Secure Interoperation in a Multidomain Environment Employing RBAC Policies

Basit Shafiq, *Student Member*, *IEEE*, James B.D. Joshi, *Member*, *IEEE Computer Society*, Elisa Bertino, *Fellow*, *IEEE*, and Arif Ghafoor, *Fellow*, *IEEE*

Abstract—Multidomain application environments where distributed multiple organizations interoperate with each other are becoming a reality as witnessed by emerging Internet-based enterprise applications. Composition of a global coherent security policy that governs information and resource accesses in such environments is a challenging problem. In this paper, we propose a policy integration framework for merging heterogeneous Role-Based Access Control (RBAC) policies of multiple domains into a global access control policy. A key challenge in composition of this policy is the resolution of conflicts that may arise among the RBAC policies of individual domains. We propose an integer programming (IP)-based approach for optimal resolution of such conflicts. The optimality criterion is to maximize interdomain role accesses without exceeding the autonomy losses beyond the acceptable limit.

Index Terms—Secure interoperation, policy integration, Role-Based Access Control (RBAC), multidomain.

1 Introduction

With the increase in information and data accessibility, there is a growing concern for security and privacy of data. Numerous studies have shown that unauthorized access, in particular by insiders, constitutes a major security problem for enterprise application environments [21], highlighting the need for robust access control management systems. This problem can get magnified in a collaborative environment where distributed, heterogeneous, and autonomous organizations interoperate with each other [13], [10]. Collaboration in such a diverse environment requires integration of all local policies to compose a global access control policy for controlling information and resource sharing across multiple domains. For secure interoperation, the interdomain data accesses supported by the global policy must be consistent with the access control policies of constituent domains. In particular, secure interoperation requires enforcement of the following two principles [10]:

- **Autonomy Principle**: If an access is permitted within an individual system, it must also be permitted under secure interoperation.
- **Security Principle**: If an access is not permitted within an individual system, it must not be permitted under secure interoperation.

- *B. Shafiq and A. Ghafoor are with the School of Electrical and Computer Engineering, Purdue University, Electrical Engineering Building, 465 Northwestern Ave., West Lafayette, IN 47907-2035. E-mail: {shafiq, ghafoor}@ecn.purdue.edu.*
- *J.B.D. Joshi is with the Department of Information Science and Telecommunications, University of Pittsburgh, 721 IS Building, 135 N. Bellefield Ave., Pittsburgh, PA 15260. E-mail: jjoshi@mail.sis.pitt.edu.*
- *E. Bertino is with the Department of Computer Sciences, Purdue University, 250 N. University St., West Lafayette, IN 47907-2066. E-mail: bertino@cerias.purdue.edu.*

Manuscript received 24 Apr. 2004; revised 27 Jan. 2005; accepted 15 May 2005; published online 19 Sept. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0119-0404.

The problem of secure interoperation has been addressed in literature in the context of multilevel security (MLS) model [10], [2]. A multilevel security or Bell-Lapadula [1] model is more suitable for environments which have static security constraints and cannot be used to capture the dynamic constraint requirements of emerging applications and information systems [13], [15]. Separation of duty (SoD) and dependence constraints are examples of such dynamic constraints and are required in most commercial applications, including digital government, E-commerce, healthcare systems, and workflow management systems [8]. The *Role-Based Access Control* (RBAC) model, due to its inherent richness in modeling hierarchical, separation of duty (SoD), cardinality, and dependency constraints, provides a promising approach to satisfy the access control requirements of the afore-mentioned applications [8], [12], [22]. Furthermore, RBAC is capable of modeling a wide range of access control policies including *discretionary access control* (DAC) and *mandatory access control* (MAC) [19].

In this paper, we propose a policy composition framework that integrates the RBAC policies of multiple domains to facilitate secure information and resource sharing in a collaborative environment. Composition of a multidomain policy governing interoperation among heterogeneous systems is a challenging task leading to various types of conflicts. These conflicts may arise because different domains may use different models, semantics, schema format, data labeling schemes, and constraints for representing their access control policies [3], [5], [6], [9], [10]. In this paper, we mainly focus on the conflicts related to access control constraints. In particular, we consider constraint conflicts arising as a result of integrating RBAC policies of multiple domains. An example of access control constraint conflict in the context of RBAC policy integration is the introduction of cycles in domain-specific role-hierarchies as depicted in Fig. 1. Such cycles in role hierarchy enable junior roles to inherit the permission of senior roles leading to violation of domain specific security constraints [10]. In addition, the interplay of role hierarchy and SoD constraints may lead to other types of constraint conflicts which are described in Section 3.2. These conflicts,

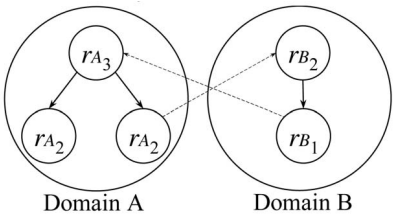

Fig. 1. An inconsistent multidomain policy because of cycles in domain-specific hierarchies.

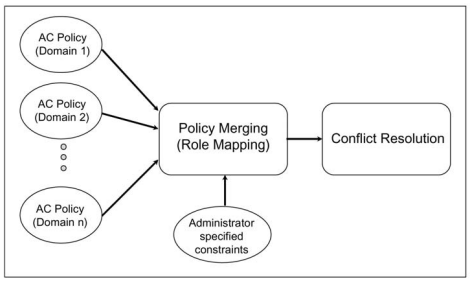

Fig. 2. Policy integration framework.

if remain undetected and unresolved, expose the collaborating systems to numerous vulnerabilities and risks pertaining to the security and privacy of their data and resources.

The proposed policy composition framework generates a secure interoperation policy in two steps as shown in Fig. 2. In the first step of policy composition, a multidomain policy is composed by defining role mapping across domains. Such a mapping enables interdomain information and resource sharing via the mapped roles. In addition to the automated generation of role mapping between cross domain roles, the framework also allows security policy administrators to map crossdomain roles based on the interoperability requirements of collaborating domains. The resulting multidomain policy may not be consistent and may not satisfy the security constraints of collaborating domains. In particular, three types of security violations, discussed in Section 3.2, may occur as a result of an inconsistent role-mapping. These include: *role-assignment violation*, *role-specific-SoD violation*, and *user-specific SoD violation*. These conflicts are resolved in the next step by removing some of the mapping links specified in the role-mapping step. Resolving policy conflicts in an arbitrary manner may significantly reduce interoperation in terms of data sharing and crossdomain accesses. The proposed policy integration framework uses an integer programming (IP)-based approach for optimal resolution of multidomain policy conflicts. The optimality criterion is to maximize information and data sharing via assumption of cross-domain roles.

An important consideration in composing an optimal interoperation policy is the preservation of domains' autonomy. Ideally, both security and autonomy of collaborating domains need to be preserved. However, satisfaction of both security and autonomy requirements may not be feasible. In almost every collaborative environment, violation of any domain's security constraints is not permissible. Domains may compromise their autonomy for establishing more interoperability provided the autonomy losses remain within the acceptable limits. The proposed IP-based approach for conflict resolution provides the flexibility of autonomy relaxation in favor of greater interoperability. Accordingly, in a collaborative environment in which certain autonomy violations can be tolerated, the objective of the conflict resolution phase is to generate an interoperation policy that maximizes interdomain role accesses and keep the autonomy losses within the acceptable limits. The main contributions of this paper are as follows:

- A novel integer programming (IP)-based approach is proposed for composition of a secure interoperation policy. The proposed approach resolves multidomain policy conflicts by finding a set of nonconflicting role-mapping links that maximizes interdomain role accesses according to the specified optimality criterion.
- The notion of *optimality* is further analyzed in terms of trade-offs between loss of autonomy and the degree of interoperation. The main feature of the second contribution is that a set of formal parameters are introduced for describing the notion of autonomy within the context of access control.

2 Related Work

The first and foremost challenge in establishing secure interoperation is the composition of a consistent and conflict-free interoperation policy that governs all the interdomain information and resource exchange. Several research efforts have been devoted to the topic of policy composition in the multidomain environment [10], [17], [5], [3]. In particular, major emphasis is given on the detection and resolution of conflicts in interoperation policies. Conflicts appearing in an interoperation policy can be divided into four types:

1. modality conflicts,
2. multiple management,
3. cyclic inheritance, and
4. separation of duties (SoD).

Modality conflicts in a policy arise because of the existence of both positive and negative authorizations for a given subject-object pair. Multiple management conflicts occur when multiple administrators having authority over a common set of subjects and objects, specify conflicting authorizations in their respective policies. In that sense, multiple management conflicts are similar to modality conflicts. Modality conflicts are resolved based on the policy precedence relationship which implies that the most specific authorization overrides the less specific one [17], [3], [9]. In case the precedence relationship cannot be established between the conflicting authorizations, the negative authorization dominates the positive one. Modality conflicts cannot occur in RBAC policy specification because negative authorizations are not supported in the RBAC model.

Cyclic inheritance conflicts mainly occur in interoperation of systems employing multilevel security policies such as lattice-based access control (LBAC) and role-based access control (RBAC) [10], [5]. In such interoperation, the cross-domain hierarchy relationship may introduce a cycle in the interoperation lattice enabling a subject lower in the access control hierarchy to assume the permissions of a subject higher in the hierarchy. Dawson et al. [5] have discussed a mediator-based framework for establishing secure interoperation among heterogeneous systems with LBAC policies. In this framework, cyclic conflicts in the interoperation lattice are resolved by the manual intervention of a policy editor. The

policy editor allows the administrator to incrementally specify the crossdomain lattice relationships. After the addition of each relationship, the editor determines the consistency of resulting interoperation policy and identifies all relationships involved in potential security violation. Interoperation conflicts are thus resolved by withdrawing all crossdomain relationships resulting in potential security violation or removing one or more relationships until the violation is corrected. Resolution of interoperation conflicts by manual intervention of policy administrator is a slow and ad hoc process and provides no guarantee on the optimality of the resulting interoperation system. In case there are multiple policy administrators, a consensus on the resolution needs to be obtained. Gong et al. [10] have investigated interoperation of systems employing multilevel access control policies. They have proposed several optimization techniques for resolution of interoperation conflicts. However, these resolution techniques are specific to cyclic inheritance conflicts and do not consider other types of interoperation conflicts.

Separation of duties (SoD) prevent two or more subjects from accessing an object that lies within their conflict of interests or disallow a subject from accessing conflicting objects or permissions, e.g., the same managers cannot authorize payments and sign the payment checks [17], [14]. Violations of SoD constraints may occur in an interoperation policy because of the interplay of various policy constraints across domains. The resolution of interoperation inconsistencies related to separation of duty constraints has not been adequately investigated and the existing approaches rely on manual intervention of policy administrators to resolve SoD conflicts [17]. As mentioned earlier, manual resolution is a tedious process and may not yield optimal interoperation. The policy composition methodology proposed in this paper provides a single framework for optimal and automated resolution of interoperation conflicts related to RBAC policies. These conflicts include both cyclic inheritance and SoD violations.

Gavrilla et al. [7] have defined a set of necessary and sufficient conditions for composition of a consistent RBAC policy. The criterion for consistent policy composition is defined in terms of cardinality, hierarchy, and SoD constraints. Accordingly, a consistent interoperation policy can be composed by incrementally checking the consistency of crossdomain role mappings. A role mapping that satisfies all the consistency conditions with respect to the resulting policy can be added to the final interoperation policy. Such incremental composition of interoperation policy depends on the order in which role mappings are evaluated and, therefore, the resulting interoperation policy may not be optimal.

3 Role-Based Access Control for Secure Interoperation

Role-based access control (RBAC) is widely used for the specification of security requirements of most commercial applications [8]. The RBAC model [22], currently being used as the basis for the NIST RBAC model, consists of the following four basic components: a set of *users*, a set of *roles*, a set of *permissions*, and a set of *sessions*. A user is a human being or a process within a system. A role is a collection of permissions associated with a certain job function within an organization. Permission defines the access rights that can be exercised on a particular object in the system. A session relates a user to possibly many roles. When a user logs in the system, the user establishes a session by activating a set of enabled roles that the user is entitled to activate at that time. If the activation request is satisfied, the user issuing the request obtains all the permissions associated with the requested roles. One of the most important aspects of RBAC is the use of role hierarchies to simplify management of authorizations. The original RBAC model supports only *inheritance* or *usage* hierarchy, which allows the users of a senior role to inherit all permissions of junior roles. In order to preserve the *principle of least privilege*, the RBAC model has been extended to include *activation hierarchy* which enables a user to activate one or more junior roles without activating senior roles [23]. A third type of hierarchy *inheritance-activation hierarchy* can be defined on roles by composing *inheritance* and *activation* hierarchies [15]. From this point onward, we will use the notations *I*, *A*, and *IA* to refer to inheritance, activation, and inheritance-activation hierarchies, respectively. The symbols $\geq_I^*$, $\geq_A^*$, and $\geq_{IA}^*$ are used to express the *I*, *A*, and *IA* relationship between two roles, respectively. Accordingly, $r_i \geq_f^* r_j$, where $f \in \{I, A, IA\}$, implies that role r_i is senior to r_j and the hierarchical relationship between them can be either *inheritance* only, or *activation* only *or inheritance-activation*. If role r_i is immediately senior to role r_j, then the superscript $*$ is omitted from the relation symbol $\geq_f$.

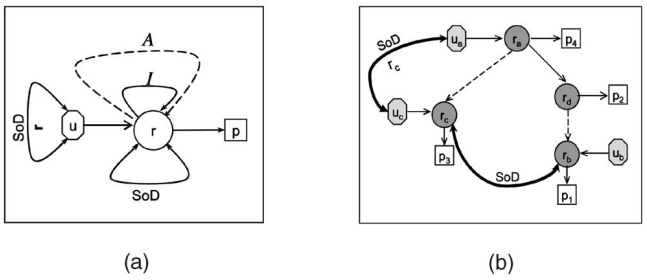

Fig. 3. (a) RBAC type graph. (b) An example of RBAC graph.

3.1 Graph-Based Specification Model for RBAC

We use a graph-based formalism similar to one described in [16] to specify the RBAC policy of a domain. In this graph-based model, users, roles, and permissions are represented as nodes and the edges of the graph describe the association between various nodes. In order to capture the RBAC semantics, the nodes cannot be connected in an arbitrary manner. The type graph, shown in Fig. 3a, defines all possible edges that may exist between different nodes. An edge between a user node u and a role node r indicates that role r is assigned to user u. Self edges on the role node r model the role hierarchy. In the type graph, *I-hierarchy* and *A-hierarchy* are represented by *solid* and *dashed edges*, respectively. There can be edges between roles and permission nodes representing permission assignments to roles. A permission is a pair (*object*, *access mode*), which describes what objects can be accessed and in which mode (read, write, execute, approve, etc.).

The graph model also supports specification of *separation of duty* (SoD) constraints. A role specific SoD constraint disallows assignment and/or activation of conflicting roles to same user. Similarly, a user specific SoD constraint prohibits conflicting users from assuming the same role

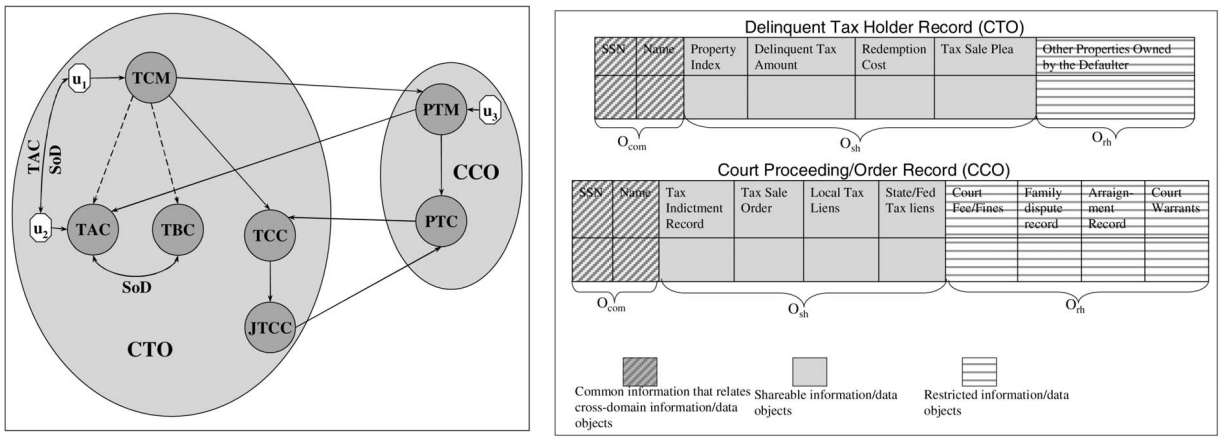

Fig. 4. (a) A multidomain access control policy defining interoperation between CTO and CCO. (b) Information exchange between CTO and CCO.

simultaneously. In the graph model, a role-specific SoD constraint between two roles is represented by a double arrow between the corresponding roles. To represent conflicting users u_i and u_j for a role r_k, a double headed edge with a label r_k is drawn between the user nodes u_i and u_j. The label r_k specifies that the corresponding users are conflicting for role r_k and cannot access r_k simultaneously (user specific SoD constraint).

Fig. 3b shows a graphical representation of an RBAC policy that consists of four roles r_a, r_b, r_c, and r_d, with $r_a \geq_A r_c$, $r_a \geq_I r_d$, and $r_d \geq_A r_b$. User u_a is assigned to r_a, u_b assigned to r_b, and u_c assigned to r_c. Note that although user u_a inherits the permissions of role r_d, it is not authorized to activate role r_b which is junior to r_d in the activation hierarchy semantics. There exists a role specific separation of duty (SoD) constraint between role r_b and r_c, shown as a double-headed arrow between these two roles in Fig. 3b. Also, users u_a and u_c are conflicting users for role r_c and are not allowed to access r_c simultaneously.

3.2 Security Requirements in a Multidomain RBAC System

The goal of policy integration is to enable information and resource sharing without violating the security of individual domains or of the multidomain system as a whole. The security and autonomy requirements of the individual domains can be extracted from their respective RBAC policies. Additional security requirements of the multi domain system can be specified by administrators with global security responsibility. The global security policy constructed from the domains' policies and administrator specified role mappings may be inconsistent and may violate the security constraints of constituent domains as well as of the multidomain system.

In this paper, we mainly focus on three types of security policy violations:

1. violation of role assignment,
2. violation of role-specific SoD constraint, and
3. violation of user-specific SoD constraint.

Definition 3.1 (role assignment violation). *An interoperation policy causes a violation of role assignment constraint of domain k if it allows a user u of domain k to access a local role r even though u is not directly assigned to r or any of the roles that are senior to r in the role hierarchy of domain k.*

Definition 3.2 (role-specific SoD violation). *An interoperation policy causes a violation of role-specific SoD constraint of domain k if it allows a user to simultaneously access any two conflicting roles r_i and r_j of domain k in the same session or in concurrent sessions.*

Definition 3.3 (user-specific SoD violation). *Let U_r^c denote the conflicting set of users for role r belonging to domain k. An interoperation policy causes a violation of user-specific SoD constraint of domain k if it allows any two distinct users from the set U_r^c to access role r in the same session or in concurrent sessions.*

The following example illustrates the three types of security violations defined above:

Example 1. Fig. 4a shows a multidomain policy that allows collaboration between *County Treasurer Office* (CTO) and *County Clerk Office* (CCO). The *County Treasure Office* has the following roles: Tax Collection Manager (TCM), Tax Assessment Clerk (TAC), Tax Billing Clerk (TBC), Tax Collection Clerk (TCC), and Junior Tax Collection Clerk (JTCC). TCM inherits all permissions of TCC which further inherits the permissions of JTCC. The roles TAC and TBC are junior to TCM in the activation hierarchy semantics, implying that a user assigned to TCM can assume the roles TAC and TBC without actually activating TCM. However, an SoD constraint is specified for TAC and TBC meaning that these roles cannot be assumed by the same user simultaneously. There is a user-specific SoD constraint between user u_1 assigned to TCM, and u_2 assigned to TAC. This SoD constraint prohibits u_1 and u_2 from assuming the role TAC concurrently. The *County Clerk Office* has only two roles, namely, Property Tax Manager (PTM) and Property Tax Clerk (PTC) with PTM inheriting the permissions of PTC. The multi-domain policy shown in Fig. 4a defines the following interoperation between CTO and CCO:

1. TCM in the *County Treasure Office* inherits all the permissions available to PTM in the *County Clerk Office*.

2. JTCC in the *County Treasure Office* inherits all the permissions available to PTC in the *County Clerk Office*.
3. PTM in the *County Clerk Office* inherits all the permissions of TAC in the *County Treasurer Office*.
4. PTC in the *County Clerk Office* inherits all the permissions of TCC in the *County Treasurer Office*.

The above multidomain policy leads to all three types of security violations. It allows JTCC to access the permissions of its senior role TCC through PTC, which is a violation of *role assignment constraint*. Moreover, this policy permits u_1 to activate roles TCM and TBC simultaneously. This leads to a violation of *role-specific SoD*, as by activating the role TCM, u_1 acquires the permissions of the role TAC through PTM. Moreover, the multidomain policy allows u_1 to activate the role TCM and u_2 to assume the role TAC. u_1 by activating TCM can inherit permissions of TAC through the role PTM. This is a violation of *user-specific SoD constraint* which prohibits u_1 and u_2 from accessing the role TAC simultaneously.

4 Multidomain Policy Composition

In this section, we elaborate on the policy merging step of Fig. 2. We first discuss the information sharing policy and the heterogeneity issues involved in composition of a multidomain policy and then we describe the general policy integration requirements that define the evaluation criterion for verifying the correctness of the composed interoperation policy. Finally, we present a policy merging algorithm RBAC-integrate that composes an interoperation policy from the RBAC policies of component domains.

4.1 Information Sharing Policy

In the policy merging step of Fig. 2, an interoperation policy is composed from the access control policies of collaborating domains. Note that a domain may not allow complete sharing of its data and resource objects. We will use the word object interchangeably for both data and resources. An object can be a file, a database relation/view, or an I/O device, etc. For each of the sharable objects, the following information needs to be provided by the controller/owner domain of that object:

1. Domains which can access the object.
2. Sanitization requirements of an object before it is shared with other domains. For instance, an object can be completely shared, or partially shared or the object cannot be shared as is, but only certain derived properties of the object are shareable (statistical information);
3. Access permissions (read, write, execute, etc.,) over an object that are available to subjects of external domains.
4. Any specific condition for sharing. For instance, an object can be shared (completely or partially) with a crossdomain subject only if a crossdomain subject has local access to certain attributes of the object in its own domain.

Fig. 4b depicts a policy for sharing *delinquent property tax information* between *County Treasurer Office* (CTO) and *County Clerk Office* (CCO). CTO maintains electronic records of tax defaulters containing information such as tax defaulters' names and social security numbers (SSN), delinquent property indices, and tax amount owed to local government, redemption cost, tax sale plea filed in district court, and details of other property/properties owned by the tax defaulter. Delinquent taxes can be sold to third parties after obtaining the tax sale order issued by the district court. The County Clerk Office (CCO), which keeps records of all court proceedings is responsible for providing the tax sale orders and other court documents related to delinquent tax holder to CTO and other concerned agencies/departments. Similarly, CCO is allowed to access the information of delinquent property, maintained by CTO, for record keeping. In order to keep privacy of personal/unrelated information, not all the information about the tax defaulter needs to be shared between the two domains. For instance, the information about other real-estate property owned by the tax defaulter is kept private and is not shared with CCO unless such property is declared delinquent. Similarly, CTO is not allowed to access any information from CCO other than tax indictment record, tax sale order, and local tax lien records. For this purpose, the tax defaulter record in the CTO is partitioned into three objects: O_{com}, O_{sT}, and O_{rT}. O_{rT} is classified information that cannot be shared with the CCO. O_{sT} is a shareable object and can be accessed by CCO. Similarly, the record in the CCO is partitioned into O_{com}, O_{sC}, and O_{rC}, where O_{rC} is confidential information, and O_{sC} can be released to CTO. The object O_{com} contains the information about the name and social security number of the defaulted person and is common to both domains. CTO can access only those records from CCO domain for which there is a corresponding O_{com} object in the delinquent tax table. Similarly, CCO can access tax/property information of only those tax holders for which the O_{com} from court records matches with the O_{com} of the delinquent tax record.

4.2 Heterogeneity Issues in Policy Composition

Composition of a global policy that governs interoperation in a multidomain system is a challenging problem. One key aspect of this complex problem is heterogeneity. There are various types of heterogeneity that need to be addressed as a part of the policy integration mechanism. The heterogeneity may arise because of naming conflicts, role hierarchies, or other policyconstraints. Naming conflicts arise because of the use of the same names to represent different conceptual entities (homonym) or different names to represent same conceptual entities (synonym). Accordingly, there may be naming conflicts among interdomain roles or objects. Naming conflicts can be resolved using schema integration techniques from the database area [11], [24]. These techniques require the use of a global lexicon to extract the conceptual meaning of attributes from their names. Additionally, domain-based and value-set-based comparisons can be performed for refinement [18]. Since a role is a collection of objects/permissions, resolving naming conflicts at the role level is difficult. Therefore, naming conflicts should be resolved at the object/permission level. Once the naming heterogeneity is resolved at the object/permission level, roles from different domains can be compared and mapped accordingly. Mapping of cross domain roles allows interoperation as explained in the next section.

In addition to naming conflicts, heterogeneity among multiple domains may exist in role hierarchies and in other dynamic constraints such as SoD and cardinality constraints. Hierarchical heterogeneity among domains' policies may arise because of two reasons: 1) use of different role hierarchies (inheritance *I*, activation *A*, inheritance-activation *IA*, hybrid [15]) by different collaborating domains;

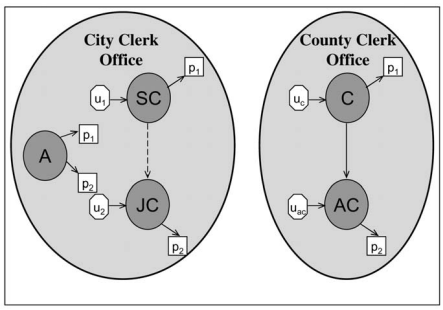

Fig. 5. Hierarchical heterogeneity.

2) domains may use different hierarchical ordering to represent same authorizations for a given role. The following example illustrates the two types of hierarchical heterogeneity that may exist between two or more cross domain roles.

Example 2. Consider the Senior Clerk (SC) and Junior Clerk (JC) roles of the City Clerk Office shown in Fig. 5a. The hierarchical relationship between SC and JC is given by A-hierarchy, $SC \geq_A JC$, i.e., SC cannot directly inherit the permissions associated with the role JC. Suppose permission p_1 is assigned to role SC and p_2 to JC. Fig. 5b shows the RBAC graph of *County Clerk Office* with two roles Clerk (C) and Assistant Clerk (AC). The Clerk role (C) inherits all the permissions of Assistant Clerk, $C \geq_I AC$. Note that the roles C and AC are assigned the same permissions as the roles SC and JC. However, roles SC and C are not equivalent because SC is not authorized for permission p_2, whereas C can directly access p_2 without activating any junior roles. The difference in authorization of the two roles is because of different types of hierarchy used in the two domains. It can also be noted in Fig. 5 that the Accountant role in the *City Clerk Office* has the same permission authorization as the Clerk role in the *County Clerk Office*, even though the hierarchical ordering for the two roles is different.

4.3 Policy Integration Requirements (PIR)

The following PIRs define the correctness semantics of an interoperation policy composed from the RBAC policies of collaborating domains. In order to be consistent with the RBAC semantics, we define the policy integration requirements using the graph-based formalism described in Section 3.1.

1. *Element preservation*: Each element (role, user, and permission) in the input RBAC graph should have a corresponding element in the multidomain graph G.
2. *Relationship preservation*: Each relationship in the input graph should be preserved in the multidomain graph G.
3. *User authorization preservation*: In the multidomain graph G, for any user u of a domain k, the authorization set of u over the objects of domain k should not be different from the authorization set specified or implied in the input RBAC policy of domain k.
4. *Order independence*: The order in which policies are integrated should not influence the output of policy integration operation.
5. *Constraint satisfaction*: The multidomain RBAC graph G must satisfy all the constraints of the input RBAC policies. In particular, no access that results in a violation of security constraints of collaborating domains can be acquired from the multidomain RBAC graph. The security constraints in RBAC policy include role assignment, role hierarchy, and SoD constraints.

The first three PIRs are important in ensuring that the authorizations of users to local resources remain unaffected in the multidomain environment and any modification in the domain policies as a result of interoperation remains transparent to the users. In particular, the access privileges of users to local resources and the access methods by which such privileges are acquired prior to interoperation should not be changed in the multidomain policy. PIR 4 entails that the final outcome of the policy integration step should not be influenced by the order in which policies are integrated. If the integration mechanism depends on the order in which policies are combined, then one must find an integration order that gives maximum interoperation with minimum overhead. However, restricting the integration order may not be an attractive option as in most collaborative environments domains join or leave collaboration any time. PIR 5 defines the security requirements of the interoperation policy in terms of the constraints of access control policies of collaborating domains. In the context of RBAC, the security constraints are defined with respect to user-role assignment, role hierarchy, and SoD constraints. All these security constraints of collaborating domains need to be preserved in the composed interoperation policy.

4.4 Merging of RBAC Policies

In this section, we focus on the issue of composing a global access control policy from the access control policies of collaborating domains. The global policy governs both intradomain and interdomain information and resource exchange. In RBAC context, integration of access control policies involves defining a mapping between crossdomain roles. *A role mapping M_{AB} is a function that maps a role of domain A to a role of domain B*$(M_{AB} : R_A \rightarrow R_B)$. By virtue of this role mapping, any user authorized for a role, say r_a, in domain A is allowed to access all the permissions of the *mapped* role, say r_b, in domain $\mathrm{B}(M_{AB}(r_a) = r_b)$.

We propose a policy merging algorithm, *RBAC-integrate*, that merges the RBAC policies of component domains by comparing and mapping crossdomain roles. The proposed policy merging algorithm finds an interdomain role mapping based on the permission assignment and hierarchical ordering of corresponding roles. The permission assignment includes both directly assigned permissions as well as inherited permissions. A permission p is a pair $p(o, a)$, where o is the object and a is the access mode. We assume that objects in the RBAC model are organized into conceptual classes, e.g., account tables, insurance claims, and audit reports, etc. Two crossdomain permissions $\mathrm{p_A} : (\mathrm{O_A}, \mathrm{a_A})$ and $\mathrm{p_B} : (\mathrm{O_B}, \mathrm{a_B})$ of domains A and B, respectively, are termed equivalent if the crossdomain objects $\mathrm{O_A}$ and $\mathrm{O_B}$ belong to the same conceptual class and the permissions $\mathrm{p_A}$ and $\mathrm{p_B}$ are declared shareable in their respective domain policies.

Using the above assumptions and the permission assignments of roles over objects, four types of relations can be defined between two crossdomain roles r_A and r_B belonging to domain A and domain B, respectively. The functions and predicates used in defining these relations are listed in Table 1.

TABLE 1
Functions/Predicates Used in the Paper

Function/predicate	Description
Pset(*r*)	Returns the set of all permissions either directly assigned to role *r* or are inherited by *r*.
$Pset_{assign}(r)$	Returns the set of permissions directly assigned to role *r*.
class(*O*)	Returns the conceptual class of object *O*.
Conf-rset(*r*)	Returns the set of all roles conflicting with role *r* i.e., roles that cannot be acquired along with role *r* by any user.
Conf-uset(*r*)	Returns the set of the sets of user that cannot acquire role *r* simultaneously.
Shareable(*O* ,*a, X*)	Returns True if permission (O, a) can be shared with domain X
Seniormost-role(G)	Returns the senior-most role of the RBAC graph G
Children(r)	Returns all roles *r'* such that $r \underset{I}{\geq} r'$ or $r \underset{A}{\geq} r'$
Common-permissions(r_1,r_2)	Returns the set of all directly assigned permissions that are common to the cross-domain roles r_1 and r_2.
Common-juniors-I(r_1,r_2)	Returns the set of roles R_j $R_j = \left\{ r : r_1 \underset{I}{\geq} r \text{ and } \exists r' \left(eq\_role(r, r') \wedge r_2 \underset{I}{\geq} r' \right) \right\}$, r_1 and r_2 are cross-domain roles.
New-role(r)	Returns True if *r* is a newly created role as a result of role splitting.
Redundant(r)	Returns True if *r* is a redundant role.
Not-compared-previously(r_1,r_2)	Returns True if the cross-domain roles r_1 and r_2 are not compared by the algorithm Role-integrate
Already-linked(r_1,r_2)	Returns True if r1 and r2 are cross-domain roles and $r_1 \underset{I}{\geq} r_2$ and $r_2 \underset{I}{\geq} r_1$
Eq_role(r_1,r_2)	Returns True if the following hold $pset_{assign}(r_1) = pset_{assign}(r_2) \wedge$ [for all r_{1j} such that $r_1 \underset{I}{\geq} r_{1j}$ there exists r_{2j} for which $r_2 \underset{I}{\geq} r_{2j}$ and $eq\_role(r_{1j}, r_{2j})$] $\wedge$ [for all r_{1j} such that $r_1 \underset{A}{\geq} r_{1j}$ there exists r_{2j} for which $r_2 \underset{A}{\geq} r_{2j}$ and $eq\_role(r_{1j}, r_{2j})$] *i.e.*, the roles r_1 and r_2 set of directly assigned permissions and are also equivalent in their hierarchical structure.
Contained(r_1,r_2)	Returns True if the following hold $\left(p \in Pset_{assign}(r_1) \Rightarrow p \in Pset_{assign}(r_2) \right) \wedge \left((r_1 \underset{I}{\geq} r_k \wedge r_k \neq r_2) \Rightarrow r_2 \underset{I}{\geq}^{*} r_k \right)$ *i.e.*, the set of directly assigned permissions of r_1 must be contained in the set of directly assigned permissions of r_2 and all the roles junior to role r_1 must also be junior to r_2 in the same hierarchy semantics.
Overlap(r_1,r_2)	Returns True if the following hold $\left(\exists p \mid p \in Pset_{assign}(r_1) \Rightarrow p \in Pset_{assign}(r_2) \right) \vee \left(\exists r_k, r_m \mid r_1 \underset{I}{\geq} r_k \Rightarrow (r_2 \underset{I}{\geq} r_m \wedge eq\_role(r_k, r_m)) \right)$
u-assign(u,r)	Returns True if user *u* is assigned role *r*.
Conf-role(r_1,r_2)	Returns True if r_1 and r_2 are conflicting roles.
Conf-user(u_1,u_2,*r*)	Returns True if u_1 and u_2 are conflicting users for role *r*.

1. *Contain*: r_A *contains* r_B if the following hold:

   a. The permission set $Pset(r_B)$ of role r_B is included in the permission set $Pset(r_A)$ of role r_A.

   $$\forall j \exists i : (O_{B_j}, a) \in Pset(r_B) \Rightarrow \left[(O_{A_i}, a) \in Pset(r_A) \wedge \left(class(O_{A_i}) = class(O_{B_j}) \right) \right].$$

   b. All the permissions in the set $Pset(r_B)$ are shareable with domain A. Formally,

   $$\forall j : (O_{B_j}, a) \in Pset(r_B) \Rightarrow shareable(O_{B_j}, a, A).$$

2. *Equivalent*: r_A is *equivalent* to r_B if r_A *contains* r_B and r_B *contains* r_A.

3. *Overlap*: r_A *overlaps* r_B if $Pset(r_A)$ and $Pset(r_B)$ have some common shareable permissions and neither r_A *contains* r_B nor r_B *contains* r_A. Formally,

   $$\begin{pmatrix} \exists i,j : class(O_{A_i}) = class(O_{B_j}) \wedge [(O_{A_i}, a) \in Pset(r_A) \wedge (O_{B_j}, a) \in Pset(r_B) \\ \wedge shareable(O_{A_i}, a, B) \wedge shareable(O_{B_j}, a, A)] \end{pmatrix} \wedge \begin{pmatrix} \neg(r_A \text{ contains } r_B) \wedge \\ \neg(r_B \text{ contains } r_A) \end{pmatrix}.$$

4. *Not related*: r_A is *not related* to r_B, if roles r_A and r_B do not share any common permissions. Formally,

   $$\neg \exists i, j : class(O_{A_i}) = class(O_{B_j}) \wedge [(O_{A_i}, a) \in Pset(r_A) \wedge (O_{B_j}, a) \in Pset(r_B)].$$

Fig. 6 shows the proposed policy merging algorithm, *RBAC-integrate*, that merges RBAC policies of n domains to produce a global multidomain policy. The input parameter G_i represents the RBAC policy of domain i specified in graphical form. This algorithm iteratively combines the RBAC policies of component domains in a pair-wise manner. In the first iteration, a composite RBAC policy is composed from domains 1 and 2 by calling the procedure, *role-integrate*, with the senior-most roles of domains 1 and 2, respectively. In the subsequent iterations, the RBAC policy of a new domain is combined with the merged RBAC policy

RBAC-integrate(G_1,G_2,…,G_n)
1. G = {V[G_1], E[G_1]}
2. **for** i ← 2 to n
3. r_1 ← seniormost-role(G)
4. r_2 ← seniormost-role(G_i)
5. G ← Role-integrate(r_1, r_2)
6. **for** each r ∈ G
7. **if** (new-role(r) **and** redundant(r))
8. **then** Remove-Role(G, r)
9. **return**

Role-integrate(r_1, r_2)
1. **for** each r_c ∈ children(r_1)
2. **do if** ((Pset(r_c) ∩ Pset(r_2) ≠ ϕ) **and** not-compared-previously(r_c,r_2))
3. **then** Role-integrate(r_c,r_2)
4. **for** each r_c ∈ children(r_2)
5. **do if** ((Pset(r_1) ∩ Pset(r_c) ≠ ϕ) **and** not-compared-previously(r_1,r_c))
6. **then** Role-integrate(r_1,r_c)
7. ►**REM** return without doing anything if r_1 and r_2 are already linked
8. **if** already-linked(r_1,r_2)
9. **then return**
10. ►**REM** contained(r_i, r_j)=True, if $\left(p \in Pset_{assign}(r_i) \Rightarrow p \in Pset_{assign}(r_j)\right) \wedge \left((r_i \geq_I r_k \wedge r_k \neq r_j) \Rightarrow r_j \geq_I^* r_k\right)$
11. **if** contained(r_2, r_1) and contained(r_1, r_2)
12. **then if** linking r_1 and r_2 do not violate RBAC consistency properties
13. **then** link(r_1, r_2)
14. **return**
15. **else if** contained(r_2, r_1)
16. **then** r_{1j}=split(r_1, common-permissions(r_1,r_2), common-juniors-I(r_1,r_2))
17. **if** linking r_{1j} and r_2 do not violate RBAC consistency properties
18. **then** link(r_{1j}, r_2)
19. **return**
20. **else if** contained(r_1, r_2)
21. **then** r_{2j}=split(r_2, common-permissions(r_1,r_2), common-juniors-I(r_1,r_2))
22. **if** linking r_1 and r_{2j} do not violate RBAC consistency properties
23. **then** link(r_{2j},r_1)
24. **return**
25. ►**REM** overlap(r_i, r_j)=True, if
$\left(\exists p \mid p \in Pset_{assign}(r_i) \Rightarrow p \in Pset_{assign}(r_j)\right) \vee \left(\exists r_k, r_m \mid r_i \geq_I r_k \Rightarrow (r_j \geq_I r_m \wedge \text{already-linked}(r_k, r_m))\right)$
26. **else if** overlap(r_1,r_2)
27. **then** r_{1j}=split(r_1, common-permissions(r_1,r_2), common-juniors-I(r_1,r_2))
28. r_{2j}=split(r_2, common-permissions(r_1,r_2), common-juniors-I(r_1,r_2))
29. **if** linking r_{1j} and r_{2j} do not violate RBAC consistency properties
30. **then** link(r_{1j}, r_{2j})
31. **return**

Fig. 6. Policy merging algorithm.

obtained in previous iteration. After $n-1$ iterations, the RBAC policies of all n domains are merged to produce a global multidomain policy. In each iteration, after calling *role-integrate*, all the newly created *redundant roles* are removed from the integrated RBAC graph. Redundant roles are created during the process of policy merging and do not have any user or permission assignment. Removal of redundant roles is essential to satisfy the order independence (PIR 4) requirement in the merged policy.

The procedure, *role-integrate*, maps interdomain roles based on their permission assignment and hierarchical ordering. *role-integrate* is a recursive algorithm that uses bottom-up strategy to establish role equivalence across two domains. The algorithm basically checks all interdomain roles for one of the above four relations. If the roles do not share any permission, then it returns without doing anything. If the interdomain roles say, r_1 and r_2, are equivalent in their permission assignment and hierarchical ordering, then they are linked together by defining a bidirectional mapping between r_1 and r_2, i.e., $r_1 \geq_I r_2$ and $r_2 \geq_I r_1$. A role mapping $r_1 \geq_I r_2$ is represented in the RBAC graph by an *I-hierarchy* edge from r_1 to r_2. Linking two interdomain roles r_1 and r_2 through a bidirectional mapping implies that a user say u_i, authorized for role r_1 inherits all the permissions of role r_2. Similarly, a user u_j authorized for role r_2 inherits all permissions in the authorization set of r_1. *Role-integrate* calls *link* function (shown in Fig. 7) for bidirectional mapping of crossdomain equivalent roles r_1 and r_2. In addition, conflicting role sets of r_1 and r_2 and all their senior roles that have an inheritance path to r_1 and r_2, and all the roles that conflict with r_1 and r_2 and their senior roles are updated in the *link* function. This update in the conflicting role sets is essential to preserve the hierarchical consistency property of the RBAC model which requires that the conflicting role set of a junior role must be contained in the conflicting role set of the senior role [7].

split(r, common-permissions, common-juniors)

1. $r_j \leftarrow$ createrole()
2. insert(r->childrenlist-I, r_j)
3. **for** each p $\in$ common-permissions
4. **do** remove(r->plist, p)
5. insert(r_j ->plist,p)
6. **for** each $r_c \in$ common-juniors
7. **do** remove(r->childrenlist-I, r_c)
8. insert(r_j ->childrenlist-I, r)
9. **return** r_j

link(r_1, r_2)

1. insert(r_1->childrenlist-I,r_2)
2. insert(r_2->childrenlist-I,r_1)
3. **for** each r_i s.t. ($r_i = r_1 \vee r_i \geq_I^* r_1$)
4. **do for** each r_j s.t. ($r_j \in$ conf-rset(r_1))$\vee$ ($r_j \geq_I^* r_c \wedge r_c \in$ conf-rset(r_1))
5. **do** conf-rset(r_i)=conf-rset(r_i)$\cup r_j$
6. conf-rset(r_j)=conf-rset(r_j)$\cup r_i$
7. **for** each r_i s.t. ($r_i = r_2 \vee r_i \geq_I^* r_2$)
8. **do for** each r_j s.t. ($r_j \in$ conf-rset(r_2))$\vee$ ($r_j \geq_I^* r_c \wedge r_c \in$ conf-rset(r_2))
9. **do** conf-rset(r_i)=conf-rset(r_i)$\cup r_j$
10. conf-rset(r_j)=conf-rset(r_j)$\cup r_i$
11. **return**

Remove-Role(r_d)

1. $R_p \leftarrow R_p \cup \{r\}$, for all r such that
2. $R_c \leftarrow R_c \cup \{r\}$, for all r such that
3. **for** each $r_p \in R_p$
4. **for** each $r_c \in R_c$
5. If $\exists r' : r' \neq r_d \wedge r_p \geq_I r' \wedge r' \geq_I r_c$
6. continue
7. insert(r_p->childrenlist-I, r_c)
8. remove(r_c->parentlist-I, r_d)
9. insert(r_p->parentlist-I, r_p)
10. **for** all r_e s.t. $r_e \in$ equivalent(r_d)
11. remove(r_e->parentlist-I, r_d)
12. remove(r_e->childrenlist-I, r_d)
13. **for** all r_s s.t. $r_d \in$ conf-rset(r_s)
14. remove(r_s->conf-rset, r_d)
15. **for** each $r_p \in R_p$
16. **for** each p $\in$ Pset(r_d)
17. insert(r_p->Pset, p)
18. deallocate(r_d)
19. **return**

Fig. 7. Procedures used by Role-Integrate during policy integration.

As a result of this update in conflicting role sets, new *SoD* constraints are added between roles which do not conflict with each other in their original domain RBAC policy. We will use the term *induced SoD constraint* to denote such *SoD* constraints that are not present in the domains' original RBAC policies. A formal definition of induced SoD constraint is given in Section 5.3.

In the presence of multiple hierarchy types, an addition of roles in the conflicting role sets may lead to a situation in which two conflicting roles, say r_2 and r_3, have a common ancestor, say r_1, which inherits both roles r_2 and r_3, (i.e., $r_1 \geq_I^* r_2, r_1 \geq_I^* r_3$). This situation can be avoided by making r_2 and r_3 conflicting roles only if they do not have a common ancestor role that inherits them. For instance, in Fig. 8b an *induced SoD constraint* is defined between roles r_2 and r_3 when these roles are mapped to conflicting roles r_4 and r_5, respectively. Both r_2 and r_3 have a common ancestor role r_1; however, r_1 does not inherit the permissions of role r_2 and r_3 as it is related to r_2 and r_3 in the *A-hierarchy* semantics.

Two crossdomain roles may also have a subset-superset (containment) or overlapping relationship. Role r_1 is contained in r_2 if the set of all permissions directly assigned to r_1 is contained in the set of permissions directly assigned to r_2, and all the roles that are junior to r_1 in the *I*-hierarchy semantics are also junior to r_2 in the *I*-hierarchy semantics. Note that the containment relation mentioned here is slightly different from the containment relation defined earlier. In this case, hierarchical ordering is also considered in addition to permission assignment in defining the containment relationship between two roles. If r_2 contains r_1, then a junior role r_{2j} is created by calling *split* function shown in Fig. 7. In the *split* function, all the permissions and junior roles (*I*-hierarchy semantics) common to both r_1 and r_2 are removed from r_2 and are assigned to r_{2j}. After permission reassignment, r_{2j} and r_1 are linked together through a bidirectional mapping, i.e., $r_1 \geq_I r_{2j}$ and

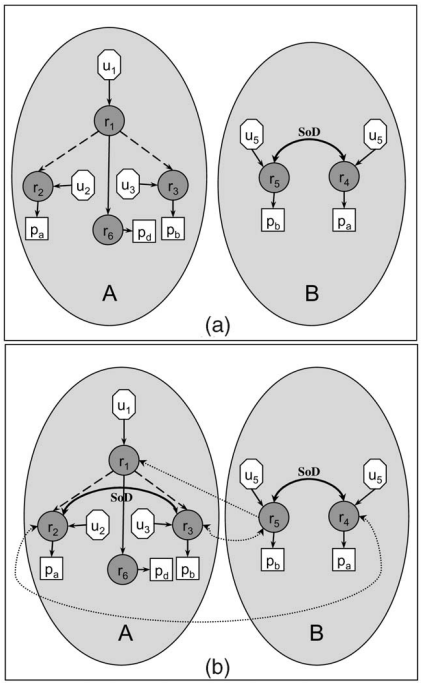

Fig. 8. (a) RBAC policy graph of domain A and B used in Example 3. (b) Integrated RBAC policy defining interoperation between domains A and B.

$r_{2j} \geq_I r_1$. If r_1 and r_2 overlap but none of the roles contain each other, then two new roles r_{1j} and r_{2j} are created and made junior to r_1 and r_2, respectively. Permissions and junior roles common to both r_1 and r_2 are removed from the senior roles r_1 and r_2 and assigned to the roles r_{1j} and r_{2j}. After this permission and role assignments, a bidirectional mapping ($r_{1j} \geq_I r_{2j}$ and $r_{2j} \geq_I r_{1j}$) is defined between r_{1j} and r_{2j}.

The policy merging algorithm, *RBAC-integrate*, runs in polynomial time and has a worst-case complexity of $O(n|P|^3)$, where n is the number of input domains and $|P|$ is the total number of permissions in the multidomain system.

In Section 6, we provide an example of an interoperation policy generated by merging the access control policies of various county offices including *County Clerk Office* (CCO), *County Treasurer Office* (CTO), and *County Attorney Office* (CAO). These county offices collaborate with each other for collection and sale of real-estate taxes on property parcels located within the jurisdiction of the concerned county. Figs. 11a, 11b, and 11c show the graphical representation of RBAC policies of CCO, CTO, and CAO domains prior to merging and Figs. 11d, 11e, and 11f depict the RBAC graph of these domains after defining crossdomain role mappings.

5 An Optimal Conflict Resolution Technique

The policy merging algorithm described above takes as input the RBAC policies of the domains and composes a multidomain policy which allows interdomain role accesses and is homogeneous in terms of role hierarchies and permission assignments. However, the multidomain policy created in this phase may be inconsistent and may not completely satisfy the collaborating domains' security requirements. Moreover, security administrator(s), in charge of the global security policy, can define additional interdomain accesses in the form of role mappings. These administrator-specified role mappings may also conflict with the access control policies of individual domains. For instance, in Figs. 11d and 11e, mapping the role *LSO* of *CCO* domain to the role *DTA* of *CTO* domain violates the role-specific SoD constraint between roles *DTA* and DTM_{10} of *CTO*. This role mapping enables user u_6 to access role *DTA* through the role *LSO*. Also, the bidirectional role mapping between $R10_{11}$ and DTM_{10} allows user u_6 to access role DTM_{10} through $R10_{11}$. This is a violation of the SoD constraint defined between roles *DTA* and *DTM* in the original RBAC policy of CTO domain shown in Fig. 11a.

The solution to this problem is to remove one of the following role mappings: 1) $LSO : CCO \geq_I DTA : CTO$ and 2) $R10_{11} : CCO \geq_I DTM_{10} : CTO$. This raises an important question: Which role mapping from the set of conflicting mappings should be removed so that the security and autonomy constraints of collaborating domains are not violated? Although, removal of crossdomain role mappings resolves conflicts in the given interoperation policy, it also changes the set of allowable accesses and an arbitrary selection of removable role mappings may significantly reduce interoperation. A conflict resolution mechanism is needed that resolves policy conflicts among the collaborating domains in an optimal manner. The problem of conflict resolution in a given multidomain RBAC policy can be formulated as an optimization problem with the objective of maximizing permitted accesses according to some prespecified optimality criterion. Various optimality measures such as maximum data sharing [10] and maximum prioritized accesses can be used.

5.1 IP Formulation of a Multidomain RBAC Policy

In the following, we describe an approach for formulating the multidomain policy integration problem into an integer program (IP) [25]. The proposed IP formulation is generic in the sense that it can work for any of the above mentioned optimality criteria. Changing the optimality measure in our formulation only requires changing the weights in the objective function.

In the IP formulation of the RBAC policy, all the constraints such as role-assignment, SoD, permitted, and restricted access constraints are defined using linear inequalities. The variables used in these inequalities convey both user and role information. For instance, the variables are of the form u_{ir_j} where the first subscript i identifies the user and the second subscript r_j specifies the role. The variable u_{ir_j} is a binary variable, i.e., it can take a value of "0" or "1" only. If the variable $u_{ir_j} = 1$, then user u_i is authorized for role r_j, otherwise u_i is not authorized for r_j and cannot access role r_j by any means. If user u_i and role r_j are from different domains and $u_{ir_j} = 0$, then in the role graph, there should not be any path from the user node u_i to the role node r_j. Note that the given multidomain RBAC policy may be inconsistent and a path may exist between user u_i from one domain and role r_j from another domain, and in the solution to the IP problem $u_{ir_j} = 0$. This inconsistency is resolved by dropping an interdomain role mapping edge that lies in the path between the user node u_i and role node r_j.

5.1.1 Constraint Transformation Rules

In the following, we list the transformation rules to generate IP constraint inequalities for an RBAC policy. In specifying the rules we denote by $\mathrm{U_k}$ and $\mathrm{R_k}$ the set of users and roles of domain k, respectively; we also denote by U the union of all $\mathrm{U_k}$s and by R the union of all $\mathrm{R_k}$s.

1. For each domain k, if a user $u_i \in \mathrm{U_k}$ is not authorized for a role $r_j \in \mathrm{R_k}$ by the access control policy of domain k, then $u_{ir_j} = 0$.
2. For a user $u_i \in \mathrm{U}$ and role $r_j \in \mathrm{R}$, if $domain(u_i) \neq domain(r_j)$ and u_i cannot inherit the permissions of role r_j, then $u_{ir_j} = 0$.
3. Let $\mathrm{A_u}$ be the set of users assigned to a role r_j. At least one user from the set $\mathrm{A_u}$ must be able to access role r_j. Formally, $\sum_{u_i \in A_u} u_{ir_j} > 0$.
4. Let $u_{ir_j} = 1$ and a role r_k exists such that $domain(r_j) = domain(r_k)$ and $r_j \geq_I r_k$, then u_i is also authorized to access role r_k, i.e., $u_{ir_k} = 1$.
5. Consider a user u_i and a role r_k such that $domain(u_i) \neq domain(r_k)$. Let $\mathrm{R_m}$ be a set of roles such that for all $r_m \in \mathrm{R_m}$, $domain(r_m) = domain(r_k)$. Also, in the RBAC graph, there is a path from u_i to r_m and $r_m \geq_I r_k$. We define two roles sets $\mathrm{R_c}$ and $\mathrm{R_{pc}}$ as follows:

$$\mathrm{R_c} = \{r | r \geq_I r_k \wedge domain(r_k) \neq domain(r)\}$$
$$\mathrm{R_{pc}} = \{r_p | \exists r \in \mathrm{R_c} \text{ such that} (r_p = r \wedge u\_assign(u, r)) \vee (r_p \geq_I r \wedge domain(r) = domain(r_p))\}.$$

The following inequalities define the conditions for a user u_i to access role r_k:

a. $\forall r_m \in \mathrm{R_m}, u_{ir_m} - u_{ir_k} \leq 0.$
b. $\sum_{r_m \in R_m} u_{ir_m} + \sum_{r_n \in \mathrm{R_c}} u_{ir_n} - u_{ir_k} \geq 0.$
c. $\sum_{r_m \in R_m} u_{ir_m} + \sum_{r_p \in \mathrm{R_{pc}}} u_{ir_p} - u_{ir_k} \geq 0.$

The above set of constraint implies that a user u_i may access a cross domain role r_k only if one of the following two conditions holds:

1. u_i is authorized for a crossdomain role r_m such that $domain(r_m) = domain(r_k)$ and $r_m \geq_I r_k$.
2. u_i is authorized for role r_n and there is an interdomain role mapping from r_n to r_k.

Condition 5c is necessary to avoid any localized assignment of 1 to variables u_{ir_k} and u_{ir_n}, where $u_{ir_n} \in \mathrm{R_c}$.

6. Consider any two users u_i and u_j and a role r_j. Suppose u_i is authorized to access role r_k, i.e., $u_{ir_k} = 1$. Suppose that a crossdomain role mapping exists from role r_k to role r_l. If user u_i is able to access r_l through the crossdomain mapping link (r_k, r_l), then user u_j, if authorized for role r_k, can also access r_l through the mapping link (r_k, r_l). Formally,

$$\begin{aligned} &\text{if } domain(u_i) = domain(u_j) = domain(r_k) \\ &\text{then } (u_{ir_k} - u_{ir_l}) - \left(u_{jr_k} - u_{jr_l}\right) = 0 \\ &\text{else } (u_{ir_k} - u_{ir_l}) - (u_{jr_k} - u_{jr_l}) \geq 0. \end{aligned}$$

7. A role specific *SoD* constraint may exist between two intradomain or interdomain roles. In the graph model, the *SoD* constraint between two conflicting roles r_j and r_k is represented by a double-headed arrow between roles r_j and r_k. In the IP formulation, this *SoD* constraint can be written as: $u_{ir_j} + u_{ir_k} \leq 1$, for all users u_i such that u_i can access either r_j or r_k.
8. Suppose that a *SoD* constraint exists between two intradomain roles r_m and r_n induced by crossdomain roles r_k and r_l. This *induced SoD* constraint can be written in equation form as: $u_{ir_m} + u_{ir_n} + u_{ir_k} + u_{ir_l} \leq 3$, for all users u_i such that u_i can access either r_m or r_n.
9. Let $\mathrm{U_{kc}}$ be the set of conflicting users for role r_k. At most, one user in the set $\mathrm{U_{kc}}$ is allowed to access or activate role r_k at any given time. Formally,

$$\sum_{u_i \in \mathrm{U}_{kc}} \mathrm{u_{ir_k}} \leq 1.$$

5.2 Optimality Criteria and Weight Assignment

The IP constraints described in the above section are used to define security requirements of collaborating domains' RBAC policies. Once the RBAC constraints are transformed into linear IP constraints by using the above transformation rules, the multidomain RBAC policy can be formulated as the following integer programming problem:

$$\begin{aligned} \text{maximize} \quad & c^T u_r \\ \text{Subject to} \quad & Au_r \leq b \\ & \forall u_{ir_j} \in u_r,\ u_{ir_j} = 0 \text{ or } 1, \end{aligned}$$

where A is the constraint matrix and c is a vector defining the optimality criteria in terms of the weight of the decision variables corresponding to user-role authorizations. The main purpose of formulating the multidomain RBAC policy into an IP problem is to find a feasible solution (a set of users to role authorization) that maximizes the objective function according to the given optimality criterion without violating the security constraints of underlying domains. Various optimality measures such as maximum data sharing and maximum prioritized accesses can be used. Maximum data sharing does not consider any priority among the interdomain accesses and involves maximizing the overall interdomain accessibility. Maximum data sharing can be specified in the objective function as a sum of all decision variables representing interdomain user to role accesses, i.e., all c_is corresponding to the crossdomain user-role variables are assigned a value of "1" and the remaining c_is are set to "0."

In some cases, certain crossdomain accesses have a higher priority than the others. Therefore, such accesses need to be assigned a higher weight for increasing their chances of retention in the final policy. The weight of a given crossdomain access is defined relative to the weights of conflicting accesses that can be removed in favor of the given access. We assume that domains may specify their preference for retention of some of the crossdomain accesses by indicating which accesses should supersede conflicting accesses. Based on this priority specification, the weights of the corresponding user-role access variables in the objective function are determined. For instance, consider the following four conflicting crossdomain accesses represented by user-role variables: u_{r1}, u_{r2}, u_{r3}, and u_{r4}. Let c_1, c_2, c_3, and c_4, respectively, denote their weights. Suppose the following rules specify the relative priorities of these accesses:

1. u_{r1} supersedes the individual accesses u_{r2}, u_{r3}, and u_{r4}, implying that either u_{r2} or u_{r3} or u_{r4} can be removed in favor of u_{r1}.
2. u_{r1} also supersedes $u_{r2} + u_{r3}$, implying that if there is a choice of retaining the single crossdomain access u_{r1} or two crossdomain accesses u_{r2} and u_{r3}, then u_{r1} is retained and both u_{r2} and u_{r3} are removed.
3. $u_{r2} + u_{r4}$ and $u_{r3} + u_{r4}$ supersede the crossdomain access u_{r1}, implying that the single crossdomain access u_{r1} can be removed in favor of joint accesses u_{r2} and r_{r4} or u_{r3} and u_{r4}.
4. u_{r4} supersedes the individual accesses u_{r2} and u_{r3}.

The weight assignment corresponding to this priority specification is given by: $c_4 > \max\{c_2, c_3\}$ and

$$\max\{c_4, c_2 + c_3\} < c_1 < (c_2 + c_3 + c_3).$$

It can be noticed that changing the weights of decision variables impact the degree of interoperability and autonomy of individual domains. In Section 6, we explain that a trade-off exist between the two metrics which depends on weight selection.

5.3 Autonomy Consideration

One key requirement of policy integration is to maintain the autonomy of all collaborating domains. However, preserving the autonomy of individual domains may significantly reduce interoperation and in some cases may not allow interoperation at all. In other words, there is a trade-off between seeking interoperability and preserving autonomy. In the RBAC policy integration framework, violation of a domain's autonomy occurs because of the following two reasons:

Induced SoD constraint: An *induced SoD* constraint is a *SoD* constraint between two intradomain roles, say r_a and r_b, which do not conflict with each other in their original domain's RBAC policy. Such a *SoD* constraint is caused by conflicting crossdomain roles, say r_c and r_d, for which the following hold:

a. $domain(r_c) \neq domain(r_a) = domain(r_b)$.
b. $domain(r_d) \neq domain(r_a) = domain(r_b)$.
c.

$$conf - rset(r_c, r_d) \wedge$$
$$[(r_a \geq_I r_c \wedge r_b \geq_I r_d) \vee (r_b \geq_I r_c \wedge r_a \geq_I r_d)].$$

Fig. 8b illustrates an *induced SoD* constraint between roles r_2 and r_3 of domain A caused by roles r_4 and r_5 of domain B. Note that in the original RBAC policy of domain A, shown in Fig. 8a, r_2 and r_3 are nonconflicting. As a result of this *induced SoD* constraint, user u_1 who in the domain A's original policy is authorized to access role r_2 and r_3 simultaneously, cannot access these roles concurrently in the multidomain system.

Asymmetric cardinality of mapped roles: There are various types of cardinalities associated with a given role, for instance, role-assignment cardinality, role-activation cardinality, per-user role-assignment cardinality, and per user role activation cardinality [15]. For simplicity of discussion, we only consider role-activation cardinality which is defined as the maximum number of concurrent accesses of a role allowed by a given RBAC policy. For a consistent RBAC policy, the cardinality of a senior role should not be greater than the cardinality of any of the junior roles that are related to the senior role in the *I*-hierarchy semantics [7]. Accordingly, a role mapping relation $r_a : A \geq_I r_b : \mathrm{B}$ between the crossdomain roles r_a and r_b of domains A and B, respectively, becomes inconsistent if the cardinality of r_a is greater than the cardinality of r_b. In order to avoid an inconsistent role mapping due to asymmetric cardinalities of mapped roles, the cardinality of the senior role in the mapping relation is reduced to the cardinality of the junior role. For instance, in the mapping relation $r_a : A \geq_I r_b : \mathrm{B}$, if r_a has a cardinality constraint of three and r_b has a cardinality constraint of one, then the cardinality of r_a is decreased to one to ensure a consistent mapping. This reduction in the role cardinality of r_a can be considered as a violation of domain A's autonomy as the number of concurrent accesses of r_a allowed in the original RBAC policy of domain A are not permitted under this interoperation policy. On the other hand, retaining the original cardinalities of interoperable roles may lead to security violations. Obviously, the third option is to disallow any crossdomain accesses via roles with asymmetric cardinalities. This option reduces interoperation between two otherwise similar crossdomain roles. Fig. 13 depicts the trade-off between interoperability and autonomy in a graphical manner. A discussion of this graph is presented in Section 6.

In general, composition of a global multidomain policy that allows interoperation among multiple domains without any violation of collaborating domains' security and autonomy is not a feasible task. In almost any collaborative environment, violation of any domain's security policy is not permissible. However, domains may be willing to compromise their autonomy for the sake of establishing more interoperability provided the autonomy losses remain within the acceptable limits. In the following, we describe how this autonomy relaxation condition can be incorporated as a constraint in the IP problem.

Let L_A denote the set of all crossdomain role mappings that either reduces role cardinalities of domain A or adds induced SoD constraints between roles of domain A. The overall autonomy loss of domain A caused by L_A is given by:

$$AL(L_A) = \frac{\begin{pmatrix}\text{Total number of local accesses without}\\ \text{any cross-domain role mapping link}\end{pmatrix} - \begin{pmatrix}\text{Total number of local accesses in}\\ \text{presence of all role-mapping}\\ \text{links in } L_A\end{pmatrix}}{\begin{pmatrix}\text{Total number of local accesses without}\\ \text{any cross-domain role mapping link}\end{pmatrix}}.$$

The above expression can also be used for computing autonomy losses of a domain caused by individual role-mappings. However, the aggregate of all link-level autonomy losses may be greater than the overall autonomy loss of a domain, i.e., $AL(L_A) \leq \sum_{l \in L_A} AL(\{l\})$. The reason for this discrepancy is that some common local accesses may be reduced by multiple crossdomain role-mapping links; therefore, reduction of these accesses is considered multiple times in the link-level aggregate. Based on the commonality of reduction of local accesses, we define a set $S_i (S_i \subseteq L_A)$ for every crossdomain role mapping link l_i such that all local accesses of domain A reduced by l_i are also reduced by each role mapping link $l_k in S_i$. In order to keep the autonomy losses of a domain within a certain threshold value, say α, the following autonomy constraint can be added in the IP problem:

$$\sum_{l_i \in L_A} \left(\prod_{l_k \in S_i, i \neq k} (1 - u_{rk}) \right) AL(\{l_i\}) u_{ri} + \sum_{(l_p, l_q \in L_A) \wedge ind\_sod(l_p, l_q)} AL(\{l_p, l_q\}) u_{rp} u_{rq} \leq \alpha.$$

The first sum in the above constraint captures the autonomy losses due to role cardinality reduction. The decision variable $u_{ri}(u_{rk})$ corresponds to the retention of crossdomain role mapping link $l_i(l_k)$, i.e., the link $l_i(l_k)$ is retained in the final policy if $u_{ri} = 1 (u_{rk} = 1)$. The term $[\prod(1 - u_{rk})] AL(l_i) u_{ri}$ implies that the role mapping link l_i causes an autonomy loss of $AL(\{l_i\})$ if no other role mapping link in the set S_i is retained in the final policy. If a role mapping link $l_k \in \mathrm{S}_i$ is retained, then the autonomy loss due to l_i is not considered in computing the overall autonomy loss because all the local accesses reduced by l_i are also reduced by l_k, implying that the autonomy loss due to the link l_i is covered by the autonomy loss due to link l_k. The second sum $\sum AL(\{l_p, l_q\}) u_{rp} u_{rq}$ in the above constraint captures the autonomy loss caused by all role mapping pairs which results in the addition of induced SoD constraints in domain A. The binary predicate *ind-sod* holds for any two crossdomain mappings l_p, and l_q if their retention in the

Maximize $u_{1r_4} + u_{1r_5} + u_{2r_4} + u_{3r_5} + 2u_{4r_2} + 2u_{5r_1} + 2u_{5r_3} + 2u_{5r_6}$

Subject to

Constraints derived from rules 1, 2, 3, and 4

c1: $u_{1r_1} = 1$, c2: $u_{1r_6} = 1$, c3: $u_{2r_1} = 0$, c4: $u_{2r_2} = 1$, c5: $u_{2r_3} = 0$, c6: $u_{2r_6} = 0$,
c7: $u_{3r_1} = 0$, c8: $u_{3r_2} = 0$, c9: $u_{3r_3} = 1$, c10: $u_{3r_6} = 0$, c11: $u_{4r_4} = 1$, c12: $u_{4r_5} = 0$,
c13: $u_{5r_4} = 0$, c14: $u_{5r_5} = 1$, c15: $u_{2r_5} = 0$, c16: $u_{3r_4} = 0$, c17: $u_{4r_1} = 0$, c18: $u_{4r_3} = 0$,
c19: $u_{4r_6} = 0$, c20: $u_{5r_2} = 0$

Constraints derived from rule 5

c21: $u_{1r_2} - u_{1r_4} \geq 0$, c22: $u_{1r_3} - u_{1r_5} \geq 0$, c23: $u_{2r_2} - u_{2r_4} \geq 0$, c24: $u_{3r_3} - u_{3r_5} \geq 0$,
c25: $u_{5r_5} - u_{5r_1} \geq 0$, c26: $u_{5r_5} - u_{5r_3} \geq 0$, c27: $u_{5r_1} - u_{5r_6} = 0$, c27: $u_{4r_4} - u_{4r_2} \geq 0$

Constraints derived from rule 6

c28: $u_{3r_3} - u_{3r_5} - u_{1r_3} + u_{1r_5} = 0$, c29: $u_{2r_2} - u_{2r_4} - u_{1r_2} + u_{1r_4} = 0$
c30: $u_{5r_5} - u_{5r_1} - u_{3r_5} + u_{3r_1} \geq 0$

Constraints derived from rule 7

c31: $u_{4r_4} + u_{4r_5} \leq 1$, c32: $u_{5r_4} + u_{5r_5} \leq 1$, c33: $u_{1r_4} + u_{1r_5} \leq 1$, c34: $u_{2r_4} + u_{2r_5} \leq 1$,
c35: $u_{3r_4} + u_{3r_5} \leq 1$, c36: $u_{1r_2} + u_{1r_5} \leq 1$, c37: $u_{2r_2} + u_{2r_5} \leq 1$, c38: $u_{4r_2} + u_{4r_5} \leq 1$,
c39: $u_{3r_3} + u_{3r_4} \leq 1$, c40: $u_{1r_3} + u_{1r_4} \leq 1$: c41: $u_{5r_3} + u_{5r_4} \leq 1$

Induced SoD Constraint derived from rule 8

c42: $u_{1r_2} + u_{1r_3} + u_{1r_4} + u_{1r_5} \leq 3$

Autonomy Relaxation constraint

c43: $16.67(u_{3r_5} u_{2r_4}) \leq 10$

Fig. 9. IP formulation of multidomain RBAC policy shown in Fig. 7.

final policy requires the addition of induced SoD constraint. The following example illustrates the formulation of IP constraints including the autonomy relaxation constraint for the multidomain RBAC policy of Fig. 8.

Example 3. Consider two collaborating domains A and B with their respective RBAC policies shown in Fig. 8a. The multidomain RBAC policy that allows interdomain accesses between A and B is shown in Fig. 8b. The bidirectional role mapping between r_3 and r_5 and the administrator-specified mapping $r_5 : \mathrm{B} \geq_I r_1 : \mathrm{A}$ that allows role r_5 to inherit permission of role r_1 makes this multidomain policy inconsistent. These two role mappings enable user u_3, assigned to the junior role r_3, to assume the senior role r_1, which is a violation of role-assignment constraint. This conflict can be resolved by either removing the role mapping $r_3 : \mathrm{A} \geq_I r_5 : \mathrm{B}$ or $r_5 : \mathrm{B} \geq_I r_1 : \mathrm{A}$. In both cases, the number of crossdomain accesses will remain the same. Note that the *SoD* constraint between r_2 and r_3 is an *induced SoD* constraint. This SoD constraint is caused by the role mappings $r_3 : \mathrm{A} \geq_I r_5 : \mathrm{B}$ and $r_2 : \mathrm{A} \geq_I r_4 : \mathrm{B}$, and reduces local accesses of domain A from six to five, causing an autonomy loss of 16.67 percent. Suppose the maximum autonomy loss allowed by domain A is 10 percent. This autonomy relaxation constraint can be specified as: $16.67(u_{3r5}u_{2r4}) \leq 10$, where $u_{3r5} = 1 (u_{2r4} = 1)$ implies that the role mapping $r_3 : \mathrm{A} \geq_I r_5 : \mathrm{B} (r_2 : \mathrm{A} \geq_I r_4 : \mathrm{B})$ is retained in the final interoperation policy. The IP formulation of the multidomain policy of Fig. 8b is shown in Fig. 9. Note that in the objective function, all the decision variables representing crossdomain role accesses are assigned a weight of one, implying that the optimality criterion is to maximize all crossdomain role accesses. An optimal solution to the IP problem shown in Fig. 9 has following values of crossdomain variable: $u_{1r_4} = 0$, $u_{1r_5} = 0$, $u_{2r_4} = 1$, $u_{3r_5} = 0$, $u_{4r_2} = 1$, $u_{5r_1} = 1$, $u_{5r_3} = 1$, and $u_{5r6} = 1$. Since $u_{3r_3} = 1$ (constraint c9 in Fig. 9), and $u_{3r_5} = 0$, the crossdomain role mapping $r_3 \geq_I r_5$ needs to be removed from the multidomain RBAC graph of Fig. 8b. Removal of the role mapping $r_3 \geq_I r_5$ also invalidates the induced SoD constraint between r_2 and r_3. Thus, the resulting multidomain policy does not cause any autonomy loss of domain A.

5.4 Conflict Resolution Algorithm

Fig. 10 shows an algorithm *ConfRes* for resolving conflicts from the RBAC graph G representing the multidomain policy. This algorithm first transforms the RBAC policy constraints into IP constraints using the rules given in Section 5.1.1. Before transforming RBAC policy constraints into IP constraints, dummy users are assigned to two classes of roles which do not have any user assigned to them. Class one includes those roles which do not have any senior role in the inheritance hierarchy semantics. The assignment of dummy users to class one roles that do not have a prior user-assignment ensures that all the roles appear in the IP constraint equations, which is essential for conflict resolution. Class two includes roles which have a nonempty set of conflicting users. The dummy user u_{dj} assigned to a class two role r_j is also included in all the conflicting sets of users for role r_j. Since u_{dj} is the only user assigned to r_j therefore $u_{djr_j} = 1$ (by transformation rule 2). This prohibits any user u_k that conflicts with u_{dj} for role r_j to inherit the permissions of r_j through a senior role r_s without activating r_j. Once all the IP constraints are defined, the IP problem is solved using the optimality criterion embedded in the objective function. Based on the solution of the IP problem, the graph G is modified by removing the conflicting crossdomain role mapping edges and the corresponding induced SoD constraints. The resulting graph defines the multidomain policy that satisfies the security requirements of all collaborating domains. This is formally proved in Section 7.

ConfRes(G)

1. Assign a dummy user u_{di} to all roles r_i for which the following hold:
   a. No user is assigned to r_i.
   b. There does not exist any role r_k for which $r_k \geq_I r_i$.
2. Assign a dummy user u_{dj} to all roles r_j which have a non-empty set of conflicting users.
3. For each role r_j that is being assigned a user u_{dj} in step 2, set $u_{djr_j}=1$ and update the conflicting set of users by doing the following:
   a. Define user-specific SoD constraint between u_{dj} and all the conflicting users for role r_j that are not assigned to r_j.
   b. add new conflicting set(s) of user for role r_j containing the dummy user u_{dj} and a user u_k for which the following holds:
   $\exists\, u_i \in U,\ r_l \in R$ such that $[(r_l \geq_A^* r_j) \wedge conf\text{-}user(u_i, u_k, r_l) \wedge (u_i \in conf\text{-}user(r_j))]$
4. Using the constraint transformation rules, write the RBAC policy constraints in algebraic form.
5. Define the objective function.
6. Find an optimal feasible solution for the integer programming (IP) problem.
7. From the multi-domain RBAC policy graph G, remove the inter-domain edge (r_i, r_j) for which there exists a user u_k such that $u_{kr_i}=1$ and $u_{kr_j}=0$ in the optimal feasible solution.
   $G = G - \{(r_i, r_j) \mid \exists u_k \in U \text{ such that } u_{kr_i}=1 \text{ and } u_{kr_j}=0 \text{ and } domain(r_i) \neq domain(r_j)\}$
8. For an edge (r_i, r_j) removed from G, if r_j induces an SoD constraint between r_i and any role r_k, then remove that induced SoD constraint from RBAC policy graph G
9. From the graph G, remove the conflicting set of users added in step 3b.

Fig. 10. Conflict resolution algorithm.

6 Illustrative Example

In this section, we illustrate the proposed policy integration framework by considering interoperation among various offices of a county for collection and sale of real-state tax on property parcels located within the jurisdiction of concerned county. The concerned county offices include: *County Clerk Office* (CCO), *County Treasure Office* (CTO), *County Attorney Office* (CAO), *District Clerk Office* (DCO), and *District Courts* (DC). These offices/departments share information among each other for budget planning, tax billing and collection, sale of delinquent taxes, auditing, and other legal purposes. Each county office keeps the information owned by it in its local databases. Integration of these local databases is needed to provide interdomain information access capability. Such an integration not only expedites the process of tax collection and sale by providing immediate access to timely, accurate, and complete information, but also improves the productivity of existing staff by reducing redundant data collection efforts among the county departments.

In order to establish interoperation among various county offices, the access control policies of the collaborating county offices need to be integrated. Due to space limitation, we only focus on interoperation among three county offices: CCO, CTO, and CAO. Table 2 lists the roles, job description, and permissions associated with each role of all three county offices. The permission authorization in Table 2 defines the access rights or permissions available to the corresponding roles on local as well as crossdomain information objects. As mentioned in Section 4.1, an information sharing policy is needed that explicitly specifies the access rights available to crossdomain roles over a local object and the conditions under which such access is granted. Table 3 shows the information sharing policy of information/data objects that can be shared among the collaborating county offices. The letters W, R, and A in the access mode columns indicate *write*, *read*, and *approve*, respectively. Note that in the information sharing policy listed in Table 3, domains that own information objects do not indicate the actual foreign domain roles that can inherit the permissions of their local objects. Rather the owner domains only specify the conditions that must be fulfilled by crossdomain roles in order to access foreign objects. Identifying the prospective crossdomain roles that can access a given object requires the knowledge of the organization hierarchy and access control policies of other collaborating domains. Acquisition of this knowledge may not be feasible as domains may not be willing to reveal their access control policies to others. It is therefore the responsibility of the policy integration mechanism to determine the roles that satisfy the condition for accessing each others information objects and map them accordingly.

The RBAC policy graphs of the county offices (CTO, CCO, and CAO) prior to role mapping are shown in Figs. 11a, 11b, and 11c. Figs. 11d, 11e, and 11f depict the policy graphs of these county offices after mapping cross domain roles. The proposed role-mapping algorithm RBAC-integrate generates a bidirectional mapping between crossdomain roles that are equivalent in their permission assignment and have similar role hierarchy. In addition, the global security policy administrator(s) may also define crossdomain role mapping for specifying interoperation requirements. In Figs. 11d, 11e, and 11f, the edge from a local role to a foreign role defines the crossdomain role mapping. A local role, in Figs. 11d, 11e, and 11f is shown as a shaded oval with solid outline, whereas a foreign role is depicted with a dashed-outlined oval. The annotations within the dashed oval describe both the names and domains of the foreign roles to which a local role is mapped. For instance, in

TABLE 2
Description of Roles Involved in Collaboration among County Offices

Role	Domain	Job Description	Permission Authorization
Treasurer	CTO	Supervises all operations of treasurer office	Inherits all permissions of TCM, TRM, and DTM
Tax Assessor (TA)	CTO	Assess/prepare tax bills	P_6, P_9, P_{10}, P_{11}
Tax Bill Approver (TBA)	CTO	Reassess & approve of tax bils	P_6, P_9, P_{10}, $P_{11,}$ P_{12}
Tax Collector (TC)	CTO	Tax collection & tax sale, record keeping of tax bidders	P_{11}, P_{13}, P_{14}, $P_{31,}$ P_{32}
Tax Collection Manager (TCM)	CTO	supervises TA, TBA, and TC	Inherits all authorized permissions of TA, TB, and TC
Tax Refund Assessor (TRA)	CTO	Assess tax refunds, prepare tax refund orders	P_6, P_9, P_{11}, $P_{17,}$ P_{18}
Tax Refund Examiner (TRE)	CTO	Reassess/approve refund orders	P_6, P_9, P_{11}, $P_{18,}$ P_{19}
Tax Refund Clerk (TRC)	CTO	Prepare refund vouchers	P_{42}, P_{43}
Tax Refund Manager (TRM)	CTO	Approve refund vouchers	P_{42}, P_{43}, P_{44}
Delinquent Tax Clerk (DTC)	CTO	Keep record of delinquent taxes	P_{11}, P_{14}, P_{20}, P_{21}
Delinquent Tax Assessor (DTA)	CTO	Assess delinquent tax records	P_{11}, P_{14}, P_{20}, P_{21},P_{22}
Delinquent Tax Manager (DTM)	CTO	Approve delinquent taxes for sale/resale (supervises DTC & DTA)	Inherit permissions of DTC & DTA, $P_{24}P_{26}P_{27}$,P_{29}, P_{31}, P_{32}, P_{34}, P_{36}
County Clerk	CCO	Supervises all operations of clerk office	Inherits all permissions of PTAM & PDTM
Property Value Assessment Officer (PVAO)	CCO	Property value assessment	P_1, P_2, P_4
Tax Assessment Clerk (TAC)	CCO	Determine property tax rates	P_2, P_4, P_5, P_6, P_9
Tax Assessment Officer (TAO)	CCO	Reassess/approve tax rates	P_2, P_4, P_6, P_7, P_9
Property Tax Assessment Manager (PTAM)	CCO	Supervise TAC & TAO	Inherits permissions of TAC & TAO
Property Indexing Officer (PIO)	CCO	Property indexing	P_2, P_3, P_4
Delinquent Taxes & Lien Officer (DTLO)	CCO	Record keeping of delinquent taxes and other tax liens	P_2, P_4, P_{11}, P_{14}, P_{21}, P_{24}, P_{27}
Lien Sale Officer (LSO)	CCO	Sale of delinquent taxes, keep record of tax buyers	Inherit permissions of DTLO, P_{28}, P_{29}, P_{30}, P_{31}, P_{32}, P_{34}, P_{36}
Redemption Cost Assessor (RCA)	CCO	Prepare redemption cost estimates for delinquent taxes	Inherit permissions of DTLO, P_{29}, P_{31}, P_{34}, P_{35}, P_{36}
Property Delinquent Tax Manager (PDTM)	CCO	Reassess/approve tax redemption cost estimates (supervises LSO & RCA)	Inherit permissions of RCA & LSO, P_{33}, P_{37}
County Attorney	CAO	Heads county attorney department	Permissions of all junior roles
Deputy County Attorney Tax Section (DCAT)	CAO	Assess/approve tax sale plea	Inherits permissions of ACAT, P_{45}
Asst. County Attorney Tax Section (ACAT)	CAO	Prepare tax sale pleas for delinquent taxes and other liens/ Supervise tax sales	Inherits permissions of PLAT, P_{25}
Para Legal tax Section (PLAT)	CAO	Keep records of information obtained from CCO & CTO for tax related affairs, assists attorneys in preparing tax sale pleas	P_2, P_4, P_6, P_9, P_{11}, P_{14}, P_{16}, P_{21}, P_{24}, P_{26}, P_{27}, P_{29}, P_{31}, P_{32}, P_{34},P_{36}

Fig. 11d, the dashed oval with annotations PLAT_{09} : CAO and ACAT:CAO represent two foreign roles PLAT_{09} and ACAT of domain CAO. A local role DTM of Domain CTO is mapped to both PLAT_{09} and ACAT as shown by the edge from DTM to the corresponding foreign roles PLAT_{09} : CAO and ACAT:CAO. The mapping from DTM to ACAT:CAO is an administrator-specified mapping as indicated by the annotation "admin." The role mapping defines inheritance relationship between crossdomain roles. For instance, in Fig. 11d, the role mapping from DTM:CTO to ACAT:CAO (DTM:CTO $\geq_I$ ACAT:CAO) implies that a user, say u_1, authorized for the local role DTM can inherit the permissions of a foreign role ACAT of domain CAO through DTM. Note that crossdomain roles are related by the *I-hierarchy* semantics only, which implies that user u_1 of CTO cannot access the permissions of role ACAT without gaining access to role DTM.

The role mappings shown in Figs. 11d, 11e, and 11f represent an inconsistent interoperation policy and do not satisfy the security requirements of the collaborating county offices. For instance, the administrator-specified mappings $TA : CTO \geq_I TAO : CCO$ (Fig. 11d) and $PIO : CCO \geq_I \mathrm{TRA} : \mathrm{CTO}$ (Fig. 11e) causes a violation of role-specific SoD constraint defined between roles TRE and TRA of CTO domain. These mappings allow user u_2 to access role TRA via the crossdomain path TA:CTO $\geq_I$ TAO:CCO $\geq_I$ PIO:CCO $\geq_I$ TRA:CTO. Moreover, u_2 by accessing the local role TA inherits the permission of TRE because of the intradomain relationship TA $\geq_I$ TRE. As a result, u_2 by accessing role TA inherits the permission of conflicting roles TRE and TRA. Similarly the role mapping DTLO:CCO $\geq_I$ DTC:CTO, LSO:CCO $\geq_I$ DTA:DTC, and $\mathrm{R10}_{11} : \mathrm{CCO} \geq_I \mathrm{DTM}_{10} : \mathrm{CTO}$ (Fig. 11e) enables user u_6 to access conflicting roles DTA and DTM_{10} of CTO domain (Fig. 11d). Note that in the original RBAC policy of CTO, an *SoD* constraint is defined between DTA and DTM (Fig. 11a). Since DTM splits into roles DTM_{10} and DTM_{12}, therefore these roles also conflict with DTA as shown in Fig. 11d. Another violation of role-specific SoD constraint between roles DTM and DTA of CTO domain occur because of the role mappings DTM:CTO

TABLE 3
Information Sharing Policy of Collaborating Domains

Information/dataObject	Owner domain	Foreign domain	Access Mode available to owner domain	Access mode available to foreign domain	Purpose of access of foreign domain	Condition for cross-domain access
Property value record (O_1)	CCO	CTO, CAO	W:P_1, R:P_2	R:P_2	Property value & tax rate assessment	Access available to subjects dealing with property tax assessment and billing
Property ownership and location record (O_2)	CCO	CTO, CAO	W:P_3, R:P_4	R:P_4	Tax billing, notification	Access available to subjects dealing with tax billing and tax auditing
Tax rate record (O_3)	CCO	CTO, CAO	W:P_5, R:P_6, A:P_7	R:P_6	Tax billing	Access available to subjects dealing with tax billing and tax auditing
Tax Bill (O_5)	CTO	CCO, CAO	W:P_{10}, R:P_{11}, A:P_{12}	W:P_{10}, R:P_{11}	Auditing, tax readjustment, imposing penalties and fines for non payment or late payment of taxes,	Access available to subjects dealing with tax billing, adjustments, refunds, tax auditing and delinquent taxes and redemption
Tax Payment record (O_6)	CTO	CCO, CAO	W:P_{13}, R:P_{14}	W:P_{13}, R:P_{14}	Auditing, receive payment in certain cases (delinquent taxes, tax/lien sale)	Access available to subjects dealing with tax billing, adjustments, refunds, tax auditing and delinquent taxes and redemption
Delinquent tax record (O_9)	CTO	CCO, CAO	W:P_{20}, R:P_{21}, A:P_{22}	W:P_{20}, R:P_{21}	Preparing tax sale plea, redemption cost estimates, tax sale, auditing	Access available to subjects dealing with delinquent taxes, tax sale, tax redemption, and tax auditing
Tax Sale Plea (O_{11})	CAO	CCO, CTO	W:P_{25}, R:P_{26}, A:P_{45}	R:P_{26}	Record keeping, identifying pending tax sales awaiting court orders, auditing	Access available to subjects dealing with delinquent taxes, tax sale, tax redemption, and tax auditing
Tax Sale Judgement Order (O_{12})	DCO	CCO, CTO, CAO		R:P_{27}	Record Keeping, tax sale and redemption, auditing	Access available to subjects dealing with delinquent taxes, tax sale, tax redemption, and tax auditing
Tax Buyer Record (O_{14})	CTO	CCO, CAO	W:P_{30}, R:P_{31}	R:P_{30}	Record Keeping, tax redemption, tax refunds, auditing	Access available to subjects dealing with delinquent taxes, tax sale, tax redemption and refunds, and auditing
Tax Redemption Record (O_{15})	CTO	CCO, CAO	W:P_{33}, R:P_{34}	W:P_{33}, R:P_{34}	Record Keeping, tax redemption and refunds, auditing	Access available to subjects dealing with delinquent taxes, tax sale, redemption (Write) and refunds, and tax auditing

$\geq_I$ ACAT:CAO (Fig. 11d), ACAT:CAO $\geq_I$ TAC:CCO (Fig. 11f), and TAC:CCO $\geq_I$ DTA:CTO (Fig. 11e). These crossdomain role mappings enable u_1 to access the conflicting roles DTM and DTA of domain CTO. A *role-assignment* violation occurs because of cyclic hierarchy created by the mappings DTA:CTO $\geq_I$ ACAT:CAO (Fig. 11d), PLAT_{09} : $\mathrm{CAO} \geq_I \mathrm{DTM} : \mathrm{CTO}$ and the intradomain hierarchy constraint ACAT $\geq_I$ PLAT $\geq_I$ PLAT_{09} of CAO domain (Fig. 11f)). This cycle in role hierarchy allows user u_4 assigned to role DTA to access the permissions of the senior role DTM. The security vulnerabilities caused by the role-mappings of Figs. 11d, 11e, and 11f are tabulated in Fig. 12.

Conflicts in the multidomain policy shown in Figs. 11d, 11e, and 11f are resolved by applying the conflict resolution algorithm *ConfRes*. *ConfRes* first transforms the RBAC policy constraints into IP constraints. This IP constraint transformation process produces almost 1,500 constraints for the multidomain RBAC policy of Fig. 11. The resulting IP problem is solved with the objective of maximizing all crossdomain accesses. The solution thus obtained removes the following crossdomain role mappings from the multidomain policy graphs of Figs. 11d, 11e, and 11f: DTM:CTO $\geq_I$ ACAT:CAO, TAC:CCO $\geq_I$ DTA:CTO, DTA:CTO $\geq_I$ ACAT:CAO, PIO:CCO $\geq_I$ TRA:CTO, and LSO:CCO $\geq_I$ DTA:CTO. A maximum of 102 crossdomain accesses are obtained if the above role mappings are removed. Note that in this case, all the crossdomain accesses are assigned equal weight in the objective function. If some crossdomain accesses are more important than others then such accesses can be prioritized by assigning them a higher weight in the objective function. This will increase the likelihood of retaining high priority accesses in the multidomain policy as discussed in Section 5.2. However, the total number of accesses cannot exceed the maximum value obtained by assigning uniform weights to all cross-domain accesses.

Figs. 13a and 13b show the trade-off between interoperability and autonomy for the domains CTO and CCO, respectively. For this analysis, interoperability of a domain is defined as a measure of the number of cross domain accesses to that domain. The autonomy losses of domains CTO and CCO for the given multidomain policy with crossdomain links $\mathrm{L_{CTO}}$ and $\mathrm{L_{CCO}}$ are determined using the AL expression given in Section 5.3. In the interoperability versus autonomy loss graph, depicted in Fig. 13, the acceptable limit for autonomy loss for both domains is set to 50 percent and the level of interoperability is varied by varying the weights of decision variables in the objective function. The maximum interoperability occurs when all cross-domain accesses have a uniform weight.

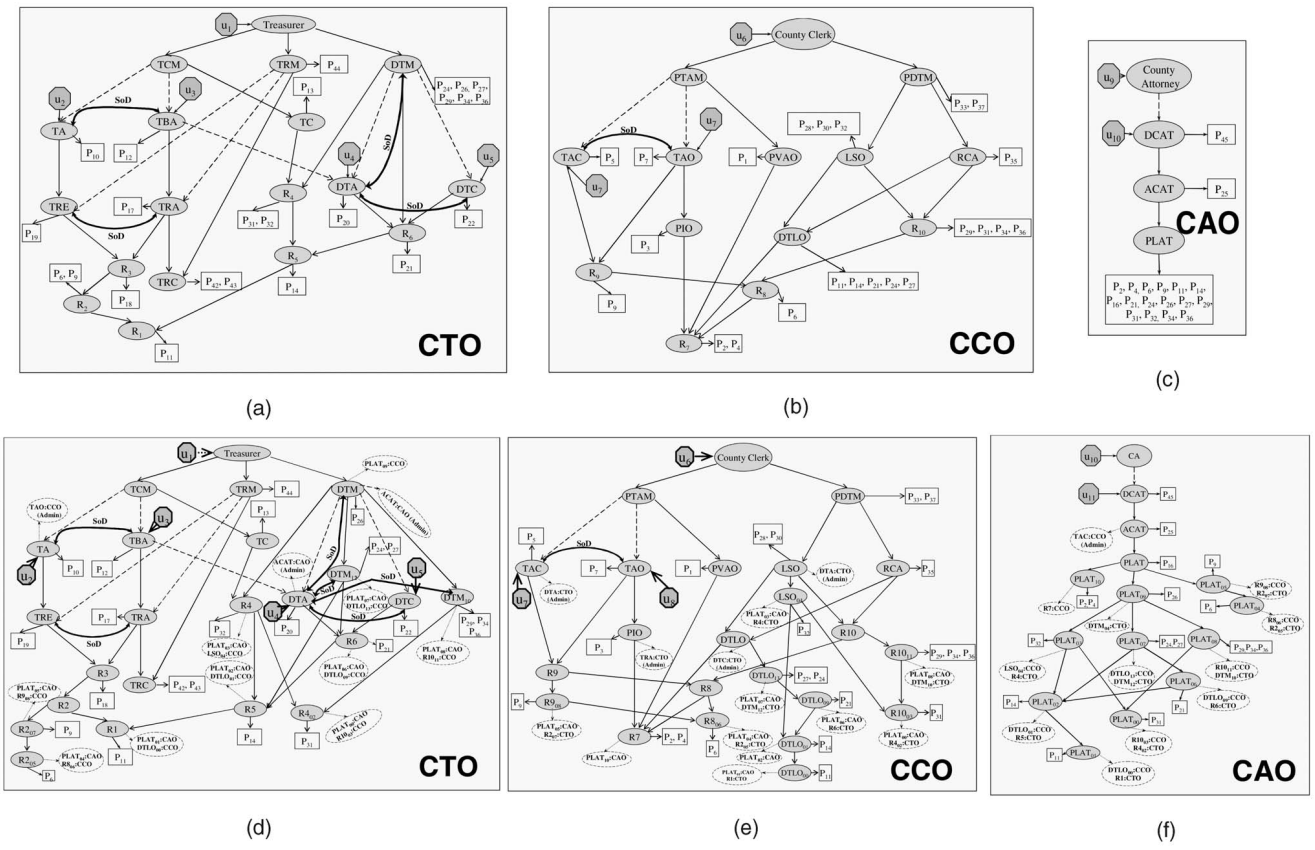

Fig. 11. (a) RBAC policy graphs of County Treasure Office (CTO), (b) of County Clerk Office (CCO), and (c) of County Attorney Office (CAO) prior to role mapping. (d) RBAC policy graphs of CTO, (e) of CCO, and (f) of CAO after role mapping.

The graph shown in Fig. 13 contains two curves defining the upper bound and lower bound for the autonomy losses at various interoperability levels. At any given interoperability level, there can be multiple values of autonomy losses corresponding to a different selection of crossdomain role-mappings. However, all the autonomy loss values are confined to the region bounded by the upper bound and lower bound curves shown in Fig. 13.

It can be noticed that the trade-off between the level of interoperation and degree of autonomy depends on the selection of weights in the objective function. Weight selection is an important issue and depends on the type of application. For example, in digital government application, achieving a high degree of interoperability among government agencies is preferable than maintaining autonomy of individual domains. In this case, uniform weights can be selected for maximizing interoperability. On the contrary, for collaborations requiring higher degree of domains' autonomy such as health-care applications, higher weights can be assigned to those crossdomain access variables that do not cause any autonomy loss. In addition, an upper bound on the autonomy loss can be specified as an additional constraint in the IP problem formulation. In summary, weight selection is an open research issue requiring further exploration.

7 Verification of Multidomain Policy

In this section, we formally analyze the proposed policy integration mechanism with respect to the five policy integration requirements (PIRs) discussed in Section 4.3. The PIRs define the correctness criteria for verifying the consistency of interoperation policy. The first four PIRs state the conformance requirements for the interoperation policy in terms of authorization preservation, relationship preservation, and order independence. The last PIR stipulates the security aspect of interoperation policy. The interoperation policy generated by the proposed policy composition framework satisfies all of the above integration requirements. To prove this claim, we first analyze the compliance of proposed framework with respect to non-security PIRs (PIR 1 - 4) and then assess the correctness of the composed interoperation policy with respect to the security constraints of collaborating domains.

7.1 Authorization and Order-independence

During the process of policy integration, the access control policies of collaborating domains may get modified; however, such modifications should not change the access privileges of local users over local objects. In addition, the integrated policy should be independent of the order in which the collaborating domains' policies are merged. These requirements are stated for RBAC policy composition in PIRs 1 - 4 in Section 4.3.

The first PIR, stipulating element preservation, holds trivially in the merged policy graph as the policy merging algorithm, *RBAC-integrate*, does not remove any element except the newly created redundant roles which are not present in the original RBAC policy graphs of collaborating domains. Similarly, all the relations specified in the original RBAC policy graphs of collaborating domains are implied in the multidomain policy graph. These relations include

Role mapping	Security Violation	Violation Type	Affected Domain
TA:CTO $\geq_I$ TAO:CCO PIO:CCO $\geq_I$ TRA:CTO	Enables u_2 to inherit the permissions of conflicting role TRE and TRA by accessing the role TA	Role-specific SoD	CTO
DTLO:CCO $\geq_I$ DTC:CTO LSO:CCO $\geq_I$ DTA:DTC $R10_{11}$:CCO $\geq_I$ DTM_{10}:CTO	Enables u_6 to access conflicting cross-domain roles DTA and DTM_{10}.	Role-specific SoD	CTO
DTM:CTO $\geq_I$ ACAT:CAO ACAT:CAO $\geq_I$ TAC:CCO TAC:CCO $\geq_I$ DTA:CTO	Enables u_1 to inherit the permissions of conflicting roles DTM and DTA concurrently.	Role-specific SoD	CTO
DTA:CTO $\geq_I$ ACAT:CAO $PLAT_{09}$:CAO $\geq_I$ DTM:CTO	Allows user u_4 assigned to role DTA to access the permissions of the senior role DTM.	Role-assignment	CTO

Fig. 12. Security violations of the multidomain policy of Fig. 11.

user-role assignment, role-permission assignments, separation of duties, and role hierarchy. The user-role assignment, permission assignment, and SoD relations remain unaltered in the multidomain RBAC graph. However, the role hierarchy and permission assignment may get changed in the process of mapping equivalent crossdomain roles. During this process, new roles may be created by splitting existing roles. As a result of this role splitting, some of the permissions assigned to the parent role, say r, may get reassigned to the newly created child role, say r_j. Also, the newly created junior role r_j may inherit the permissions of some of the roles junior to the parent role r in the *I-hierarchy* semantics. However, the *I-hierarchy* relation $r \geq_I r_j$ preserves all the hierarchy relationships between the parent role r and all its junior roles. This means that all the permissions that can be acquired through r prior to role splitting can also be acquired after splitting of r. Hence, the user authorizations specified in the original RBAC policies of collaborating domains are preserved in the multidomain policy graph.

To prove that the composed interoperation policy is independent of the order in which domains' policies are merged, we need to show that the policy integration algorithm, *RBAC-integrate*, is both commutative and associative. The proof of the commutativity and associativity properties of *RBAC-integrate* is given in the Appendix which can be found on the Computer Society Digital Library at http://www.computer.org/tkde/archives.htm.

7.2 Security Constraints

In this section, we formally prove that the interoperation policy composed by the proposed policy integration mechanism is secure. In particular, we show that no security vulnerability due to role-assignment violation (Definition 3.1), role-specific SoD violation (Definition 3.2), and user-specific SoD violation (Definition 3.3) can occur in the interoperation policy. Note that in the context of RBAC, these are the only three security vulnerabilities that may lead to unauthorized accesses. The consistency conditions defined in [7] also check the correctness of RBAC policy specification against the violation of the above three constraints.

7.2.1 Notations

For stating the above claim about security of interoperation policy in a formal manner, a state-based representation is needed. Since it is difficult to comprehend the state-transition semantics of RBAC policy from the graph-based specification, we introduce some matrix-based notations and definitions for specifying the security properties of interoperation policy.

Let $\mathrm{A_k}$ denote the adjacency matrix corresponding to the RBAC graph (with only user-role nodes) of domain k and $\mathrm{A_k}^+$ be the transitive closure of adjacency matrix $\mathrm{A_k}$. $\dim(\mathrm{A_k}) = \dim(\mathrm{A_k^+}) = (|\mathrm{U_k}| + |\mathrm{R_k}|) \times (|\mathrm{U_k}| + |\mathrm{R_k}|)$, where $\mathrm{U_k}$ is the set of user and R_k is the set of roles of domain k. The authorization of users over roles can be determined by applying the projection operator π_{ur} over the corresponding closure matrix.

Projection operator: A projection operator π_{ur} takes an adjacency or closure matrix as input and returns a matrix with users along the rows and roles along the column. $\pi_{ur} : \{\mathrm{A_k}, \mathrm{A_k}^+\} \rightarrow \mathrm{U_k} \times \mathrm{R_k}$. Projection of a closure matrix $\mathrm{A_k}^+$ defines all possible user to role authorizations in domain k:

$$\forall a_{ij} \in \pi_{ur}(\mathrm{A_k^+}), a_{ij} = \begin{cases} 1, & \text{if there is an access path from } u_i \text{ to } r_j \\ 0, & \text{otherwise.} \end{cases}$$

Note that *a SoD* or cardinality constraint may prevent u_i from accessing r_j even though $a_{ij} = 1$ in the projected closure matrix.

State matrix: A state matrix S is a matrix of dimension $|\mathrm{U}| \times |\mathrm{R}| (\mathrm{U} = \bigcup_k \mathrm{U_k}, \mathrm{R} = \bigcup_k \mathrm{R_k})$ and it describes the user to roles accesses in the multidomain environment. Note

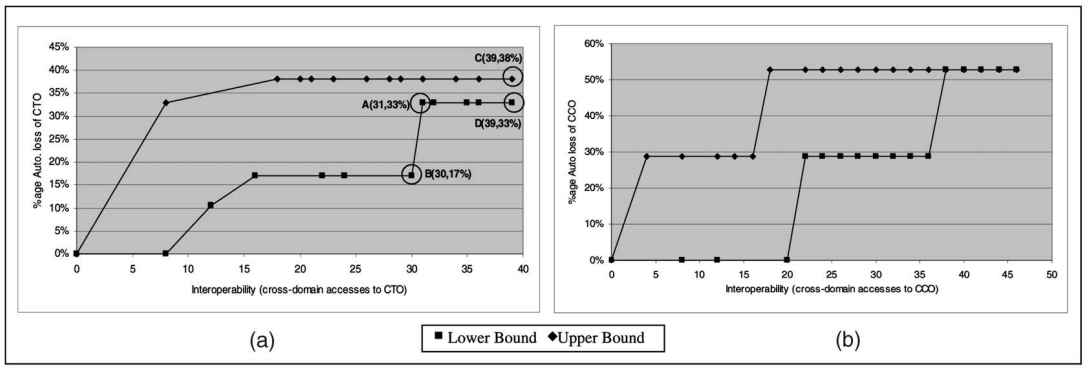

Fig. 13. Interoperability versus autonomy loss.

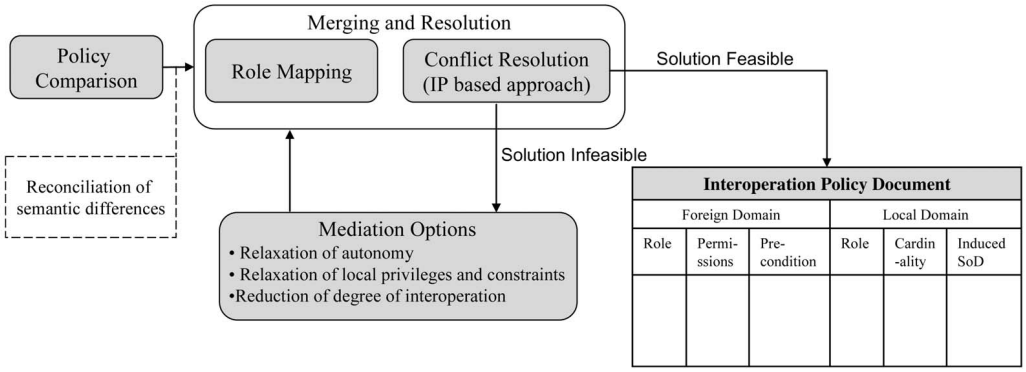

Fig. 14. Overall process of policy integration and mediation.

that the state matrix captures both intradomain and interdomain role accesses. For any $s_{ij} \in \mathrm{S}$,

$$s_{ij} = \begin{cases} 1, & \text{role } r_j \text{ is being accessed by user } u_i \\ 0, & \text{otherwise.} \end{cases}$$

7.2.2 Verification of Security Constraints

Any access control state derived from the multidomain RBAC policy is secure if it does not violate the security constraints of collaborating domains' RBAC policies. This is formally stated in the following definition:

Definition 7.1. *A state* S *is secure with respect to the* role-assignment, role-specific SoD, *and* user-specific SoD *constraints of domain* k *if and only if following conditions hold in* S*:*

1. $\forall\, a_{ij} \in \pi_{ur}(\mathrm{A}_k^+)$ *and* $s_{ij} \in \mathrm{S}, s_{ij} \leq a_{ij}$.
2. *There does not exist any user* $u_i \in \mathrm{U}$ *who accesses two or more roles in the conflicting role set*

$$R_{con} = \{r_1, \ldots, r_n | conf\text{-}role(r_i, r_j), \\ 1 \leq i, j \leq n \text{ and } i \neq j\}.$$

Formally,

$$\forall u_i \in \mathrm{U}, \sum_{r_j \in R_{con}} s_{ij} \leq 1.$$

3. *Let* U_{r_con} *be the set of conflicting user sets* (λ_r) *of role* r. $\mathrm{U}_{r\_con} = \{\bigcup_m \lambda_r^m\}$ *and*

$$\lambda_r^m = \{u_{m_1}, \ldots, u_{m_p} | conf\text{-}user(u_{m_i}, u_{m_j}, r), \\ 1 \leq i, j \leq p \text{ and } i \neq j\}.$$

For each role $r \in \mathrm{R_k}$ *which have a nonempty set* U_{r_con}*, at most one user from each of the conflicting user sets* ($\lambda_r \in \mathrm{U}_{r\_con}$) *accesses role* r *in state S. Formally,*

$$\forall r_j \in \mathrm{R}_k \left(\forall \lambda \in \mathrm{U}_{r\_con}, \sum_{u_i \in \lambda} s_{ij} \leq 1 \right).$$

The first condition in the above definition captures the role assignment constraint, i.e., in any secure state a user can access a local role if and only if there is an intradomain access path from the user node to the role node in the local access control policy of corresponding domain. The second condition specifies that conflicting roles cannot be accessed by same user in any secure state and the third condition defines the user-specific SoD constraint implying that conflicting users of a role cannot access that role concurrently in any secure state.

Having defined the necessary and sufficient conditions for a secure access control state, we claim in the following theorem that the interoperation policy generated by the proposed policy integration framework is secure. In particular, any state that can be derived from the interoperation policy will not cause any violation of role-assignment, role-specific SoD, and user-specific SoD constraints specified in the local RBAC policies of collaborating domains.

Theorem 7.2. *Given* $\mathrm{G}_1, \ldots, \mathrm{G_n}, \mathrm{n} \geq 2$*, the RBAC policy graphs of* n *collaborating domains. Let* G *be the multidomain RBAC graph composed from* $\mathrm{G}_1, \ldots, \mathrm{G_n}$ *by applying the role-mapping algorithm,* RBAC-integrate*, and conflict resolution algorithm,* ConfRes*. Assuming all* $\mathrm{G_i}$s *are consistent and conflict-free, any state* S *reachable from the multidomain policy graph* G *is secure with respect to the* role-assignment, role-specific SoD, *and* user-specific SoD *constraints defined in each* $\mathrm{G_i}(1 \leq i \leq n)$.

The proof of Theorem 7.2 is given in the Appendix which can be found on the Computer Society Digital Library at http://www.computer.org/tkde/archives.htm.

8 Multidomain Policy Integration and Mediation Process

In this section, we describe an overall process of policy integration and mediation. The process, shown in Fig. 14, consists of following phases: *policy comparison*, *merging and resolution*, and *policy mediation*.

The policy comparison phase deals with the reconciliation of semantic differences among the access control policies of collaborating domains. In this phase, domains' access control policies are analyzed to identify shareable crossdomain objects and to resolve the semantic conflicts among these objects. The technical challenges related to the resolution of semantic heterogeneity are discussed in Sections 4.2 and 4.3. We have not described any specific strategies for resolution of semantic heterogeneity as it is beyond the scope of this paper. However, this issue has been extensively investigated by the database community [11], [18], [20], [24], [26].

In the merging and resolution phase, the access rights of users over the crossdomain objects are established and the resulting authorization conflicts are resolved. In the context of RBAC, interdomain authorizations are defined by mapping crossdomain roles. Role mappings can be established automatically based on the correspondence among crossdomain shareable objects as discussed in Section 4.4. In addition, the security policy administrators, responsible for global interoperation policy, may also specify such mapping. For resolution of authorization conflicts due to inconsistent role mapping, the IP-based approach discussed in Section 5, can be used. The solution to the underlying IP problem may not be feasible implying that a multidomain policy with the given security, autonomy, and interoperability constraints cannot be composed. In this case, a new multidomain policy needs to be composed after revising the local policies and interoperability constraints. Such revision may include relaxation of autonomy requirements, relaxation of local privileges and constraints, and reduction in the degree of interoperability. The revised policies and interoperability constraints are analyzed in the mediation phase and need to be approved by the respective domains' policy administrators. For automating the mediation process, the policy administrators may specify the possible policy relaxations a priori, in decreasing order of preference, along with the acceptable bounds or thresholds on such relaxations.

If a feasible solution to the IP problem exists, the IP-based conflict resolution module generates a consistent and secure multidomain policy with maximal interoperation support under the given optimality measure and interoperation constraints. For analyzing the implications of the resulting policy, a global policy document can be generated from the composed multidomain policy for each collaborating domain. A possible schema for such document is shown in Fig. 14. This document facilitates a domain policy administrator in assessing the degree of interoperability and the level of autonomy offered by the composed multidomain policy. For instance, the document shown in Fig. 14 contains information about the crossdomain roles that can be accessed by local users of a domain, the permissions associated with the accessible crossdomain roles, and the preconditions for accessing shareable crossdomain roles. The precondition may specify what local role a user must assume before accessing a crossdomain role. In addition, the implications of the composed policy with respect to a domain's autonomy can be assessed from the local domain subschema of the policy document of Fig. 14. This subschema specifies the local roles with reduced cardinalities or local roles with induced SoD constraints. As mentioned above, both reduction in role cardinalities and addition of induced SoDs, amount to autonomy loss for a given domain.

9 Conclusion

In this paper, we have addressed the issue of secure interoperation in a multidomain environment. In particular, we focused on the problem of integrating the access control policies of heterogeneous and autonomous domains to allow interdomain information and resource sharing in a secure manner. The proposed policy integration mechanism is a two-step process including composition of a global multidomain policy from the access control policies of collaborating domains and removing conflicts from the global policy in an optimal manner. Another key requirement of policy integration is to maintain the autonomy of all collaborating domains. However, there is a trade-off between seeking interoperability and preserving autonomy. Violation of a collaborating domain's security policy in general is not permissible. However, domains may tolerate some autonomy loss for establishing more interoperability. In this paper, we have formulated the problem of secure interoperation as an optimization problem with an objective of maximizing interoperability without causing any security violation of collaborating domains and keeping the autonomy losses within acceptable limits.

The exponential complexity of the IP problem for optimal solution is a major concern in dynamic collaborative environments that have real-time constraints for policy composition. The security and access control requirements in such collaboration may change because of the following reasons:

1. evolution of collaborating domains' access control policies,
2. addition of new domains in the collaborative system, and
3. removal of domains from collaboration.

To incorporate the new security requirements, the access control policies need to be reintegrated by invoking the IP problem with new constraints. Various approximation algorithms such as Lagrangian relaxation, tabu search, and simulated annealing can be used to solve the underlying IP problem for near optimal solution in polynomial time. Another option to reduce the excessive computation overhead of policy integration is to incrementally resolve policy conflicts in an iterative manner, i.e., policies are integrated by resolving conflicts between two domains and the resulting policy is integrated with the policy of the next collaborating domain and so on. However, this iterative scheme may not yield optimal resolution because the role mapping links that are removed in a previous conflict resolution iteration will not be considered in the next iteration. Consequently, the solution in the iterative scheme is searched in a space smaller than the search space of the global conflict resolution scheme. Studying the performance trade-off between these heuristics and the global conflict resolution scheme is an interesting problem that needs further research considerations.

Acknowledgments

This research was supported by the US National Science Foundation award IIS-0209111 and by the Center for Education and Research in Information Assurance and Security at Purdue University.

References

[1] D. Bell and L. Lapadula, "Secure Computer Systems: Mathematical Foundations," Technical Report MTR-2547, vol. 1, MITRE Corp., Mar. 1973.

[2] P.A. Bonatti, M.L. Sapino, and V.S. Subrahmanian, "Merging Heterogeneous Security Orderings," *Proc. European Symp. Research in Computer Security (ESORICS)*, pp. 183-197, 1996.

[3] P. Bonatti, S.D.C. Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," *ACM Trans. Information and System Security*, vol. 5, no. 1, Feb. 2002.

[4] E. Cohen, R.K. Thomas, W. Winsborough, and D. Shands, "Models for Coalition-Based Access Control," *Proc. Seventh ACM Symp. Access Control Models and Technologies*, June 2002.

[5] S. Dawson, S. Qian, and P. Samarati, "Providing Security and Interoperation of Heterogeneous Systems," *Distributed and Parallel Databases*, vol. 8, pp. 119-145, Aug. 2000.

[6] X. Qian and T.F. Lunt, "A MAC Policy Framework for Multilevel Relational Databases," *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 1, pp. 3-15, Feb. 1996.
[7] S.I. Gavrila and J.F. Barkley, "Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management," *Proc. Third ACM Workshop Role-Based Access Control,* Oct. 1998.
[8] E. Bertino, E. Ferrari, and V. Atluri, "The Specification and Enforcement of Authorization Constraints in Workflow Management Systems," *ACM Trans. Information and System Security,* vol. 2, no. 1, pp. 65-104, 1999.
[9] E. Bertino, F. Buccafurri, E. Ferrari, and P. Rullo, "A Logical Framework for Reasoning on Data Access Control Policies," *Proc. 12th IEEE Computer Security Foundations Workshop,* pp. 175-189, 1999.
[10] L. Gong and X. Qian, "Computational Issues in Secure Interoperation," *IEEE Trans. Software Eng.,* vol. 22, no. 1, Jan. 1996.
[11] G. Yan, W.K. Ng, and E. Lim, "Product Schema Integration for Electronic Commerce–A Synonym Comparison Approach," *IEEE Trans. Knowledge and Data Eng.,* vol. 14, no. 3, pp. 583-598, May/June 2002.
[12] G.J. Ahn, M. Kang, J. Park, and R. Sandhu, "Injecting RBAC to Secure a Web-Based Workflow System," *Proc ACM Workshop Role-Based Access Control (RBAC),* 2000.
[13] J.B.D. Joshi, A. Ghafoor, W. Aref, and E.H. Spafford, "Digital Government Security Infrastructure Design Challenges," *Computer,* vol. 34, no. 2, pp. 66-72, Feb. 2001.
[14] J.B.D. Joshi, E. Bertino, and A. Ghafoor, "Temporal Hierarchies and Inheritance Semantics for GTRBAC," *Proc. Seventh ACM Symp. Access Control Models and Technologies,* pp. 74-83, June 2002.
[15] J.B.D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "Generalized Temporal Role Based Access Control Model," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 1, pp. 4-23, Jan. 2005.
[16] M. Koch, L.V. Mancini, and F.P. Presicce, "A Graph-Based Formalism for RBAC," *ACM Trans. Information and System Security,* vol. 5, no. 3, pp. 332-365, Aug. 2002.
[17] E. Lupu and M. Sloman, "Conflicts in Policy-Based Distributed Systems Management," *IEEE Trans. Software Eng.,* vol 25, no. 6, pp. 852-869, Nov. 1999.
[18] W.S. Li and C. Clifton, "Semantic Integration in Heterogeneous Databases Using Neural Networks," *Proc. Very Large Data Bases Conf.,* 1994.
[19] S.L. Osborn, R. Sandhu, and Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Trans. Information and System Security,* vol. 3, no. 2, pp. 85-106, Feb. 2000.
[20] R. Pottinger and P.A. Bernstein, "Merging Models Based on Given Correspondences," *Proc. Very Large Data Bases Conf.,* pp. 826-873, 2003.
[21] R. Power, *'Tangled Web': Tales of Digital Crime from the Shadows of Cyberspace.* Que/Macmillan Publishing, Aug. 2000.
[22] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, "Role Based Access Control Models," *Computer,* vol. 29, no. 2, Feb. 1996.
[23] R. Sandhu, "Role Activation Hierarchies," *Proc. Third ACM Workshop Role-Based Access Control,* Oct. 1998.
[24] V. Vet and N. Mars, "Bottom-Up Construction of Ontologies," *IEEE Trans. Knowledge and Data Eng.,* vol. 10, no. 4, pp. 513-526, July/Aug. 1998.
[25] L.A. Wolsey, *Integer Programming.* New York: John Wiley, 1998.
[26] C. Batini, M. Lenzrini, and S.B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys,* vol. 18, no. 4, pp. 323-364, 1986.

Basit Shafiq received the BS degree in electronic engineering from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan, in 1999, and the MS degree in electrical and computer engineering from Purdue University, West Lafayette, Indiana, in 2001. He is a PhD candidate in the School of Electrical and Computer Engineering at Purdue University and is expected to graduate in December 2005. His research interests include information security, access control management in distributed systems, and multimedia information systems. He is a student member of the IEEE.

James B.D. Joshi received the BE degree in computer science and engineering from Motilal Nehru Regional Engineering College, Allahabad, India, in 1993, the MS degree in computer science from Purdue University in 1998, and the PhD degree in computer engineering from the School of Electrical and Computer Engineering at Purdue University in 2003. He is an assistant professor in the Department of Information Sciences and Telecommunications, School of Information Sciences, at the University of Pittsburgh. Dr. Joshi is a coordinator of the Laboratory of Education and Research in Security Assured Information Systems (LERSAIS) at the University of Pittsburgh. His research interests are information systems security, database security, distributed systems, and multimedia systems. He is a member of the ACM and the IEEE Computer Society.

Elisa Bertino is a professor of computer science and of electrical and computer engineering at Purdue University and serves as a research director of CERIAS. She is also a faculty member in the Department of Computer Science and Communication at the University of Milan where she is the director of the DB and SEC laboratory. Her main research interests include security, privacy, database systems, object-oriented technology, and multimedia systems. In those areas, Dr. Bertino has published more than 250 papers in all major refereed journals, and in proceedings of international conferences and symposia. Her research has been funded by several entities and organizations both in USA and Europe, including the US National Science Foundation and the Italian Telecom. She is a coauthor of several books, a coeditor in chief of the *Very Large Database Systems (VLDB) Journal* and a member of the advisory board of the *IEEE Transactions on Knowledge and Data Engineering*. She serves also on the editorial boards of several scientific journals. She has been consultant to several Italian companies on data management systems and applications and has given several courses to industries. She is involved in several projects sponsored by the EU. She has served as a program committee member of several international conferences and as program chair of the 2004 Extending Database Technology (EDBT 2004) Conference. She is a fellow of the IEEE and a fellow of the ACM and has been been named a Golden Core Member for her service to the IEEE Computer Society. She received the 2002 IEEE Computer Society Technical Achievement Award for "outstanding contributions to database systems and database security and advanced data management systems."

Arif Ghafoor is currently a professor in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, Indiana, and is the director of the Distributed Multimedia Systems Laboratory and Information Infrastructure Security Research Laboratory. He has been actively engaged in research areas related to database security, parallel and distributed computing, and multimedia information systems and has published extensively in these areas. Dr. Ghafoor has served on the editorial boards of various journals including *ACM/Springer Multimedia Systems Journal*, the *Journal of Parallel and Distributed Databases*, and the *International Journal on Computer Networks*. He has served as a guest/coguest editor for various special issues of numerous journals including *ACM/Springer Multimedia Systems Journal*, the *Journal of Parallel and Distributed Computing*, *International Journal on Multimedia Tools and Applications*, *IEEE Journal on the Selected Areas in Communications*, and the *IEEE Transactions on Knowledge and Data Engineering*. He has coedited the book *Multimedia Document Systems in Perspectives* and has coauthored the book *Semantic Models for Multimedia Database Searching and Browsing* (Kluwer Academic Publishers, 2000). He is a fellow of the IEEE. He has received the IEEE Computer Society 2000 Technical Achievement Award for his research contributions in the area of multimedia systems.